

Gaussian Robust Classification

A thesis submitted in partial fulfillment of the
requirements for the degree of Master of Science

by
Ido Ginodi

Supervised by Dr. Amir Globerson

December 2010

The School of Computer Science and Engineering
The Hebrew University of Jerusalem, Israel

Abstract

Supervised learning is all about the ability to generalize knowledge. Specifically, the goal of the learning is to train a classifier using training data, in such a way that it will be capable of classifying new unseen data correctly. In order to achieve this goal, it is important to carefully design the learner, so it will not *overfit* the training data. This can be done in a couple of ways, where adding a regularization term is probably the most common one. The statistical learning theory explains the success of the regularization method by claiming that it restricts the complexity of the learned model. This explanation, however, is rather abstract and does not have a geometric intuition.

The generalization error of a classifier may be thought of as correlated with its robustness to perturbations of the data. Namely, if a classifier is capable of coping with disturbance, it is expected to generalize well. Indeed, it was established that the ordinary SVM formulation is equivalent to a robust formulation, in which an adversary may displace the training and testing points within a ball of pre-determined radius (Xu et al. [2009]).

In this work we explore a different kind of robustness. We suggest changing each data point with a Gaussian cloud centered at the original point. The loss is evaluated as the expectation of an underlying loss function on the cloud. This setup fits the fact that in many applications, the data is sampled along with noise. We develop a robust optimization (RO) framework, in which the adversary chooses the covariance of the noise. In our algorithm named GURU, the tuning parameter is the variance of the noise that contaminates the data, and so it can be estimated using physical or applicative considerations. Our experiments show that this framework generates classifiers that perform as well as SVM and even slightly better in some cases. Generalizations for Mercer kernels and for the multiclass case are presented as well. We also show that our framework may be further generalized, using the technique of *convex perspective* functions.

Contents

1	Introduction	5
1	Motivation	5
2	The supervised learning framework	5
3	Support Vector Machines	7
4	Robustness	8
5	Between robustness and regularization	9
6	Our contribution	10
7	Related work	10
2	Gaussian Robust Classification	12
1	Problem Formulation	12
2	The adversarial Choice	13
2.1	The structure of the loss function	13
2.2	The optimal covariance matrix subject to a trace constraint	16
3	A smooth loss function	17
4	GURU: a primal algorithm	22
5	Experiments	23
3	Dual formulation	27
1	Mathematical Derivation	27
2	A general framework	33
4	Introducing Kernels	37
1	A representer result	37
2	KEN-GURU: A primal kernelized version of GURU	40
3	Experiments	43
5	The Multiclass Case	45
1	Problem formulation	45
2	The adversarial choice	46
2.1	Applying a spectral norm constraint	46
2.2	The connection to the trace constraint	48
3	M-GURU: a primal algorithm for the multiclass case	49

4	Experiments	50
6	Discussion	52
1	Contribution	52
2	Generalizations	53
A	Single-Point Algorithms	57
1	Problem presentation	57
2	Computing the optimal displacement	58
3	ASVC: Adversarial Support Vector Classification	58
4	The Multiclass Case	59
5	Related work	60
B	Diagonal Covariance	61
C	Using the Multi-Hinge Loss	63

Chapter 1

Introduction

1. Motivation

The ability to understand new unseen data, based on knowledge that was gained using a training sample, is probably the main goal of machine learning. In the supervised learning setup, one is given a training set, consists of data samples along with labels indicating their 'type' or 'class'. The learning task in this case is to develop a decision rule, which will allow predicting the correct label of unfamiliar data.

As the main goal is to be able to generalize, it makes sense to design the learning process so it reflects the conditions under which the classifier is going to be tested and used. In many real world applications, the data we are given is corrupted by noise. The noise may be either inherent to the process that generates the data or adversarial. Examples to an inherent noise include a noisy sensor and natural variability of the data. Adversarial noise is present for example in spam emails. In either way, it is vital to learn how to classify when it is present. We suggest to do it by preparing for the worst case. Amongst all noise distribution that have a bounded power (i.e. bounded covariance), the Gaussian noise is believed to be the most problematic, since it has the maximal entropy.

By designing a classifier that is robust to Gaussian noise, we are able to learn and generalize well, without the need to introduce an explicit regularization term. In that respect, our work aims at shading more light on the connection between robustness and generalization.

2. The supervised learning framework

Formally speaking, the supervised learning setup consists of three major components:

1. **Data.** We denote \mathcal{X} the sample space, in which the data samples live (i.e. the objects one tries to classify. e.g., vector representation of handwritten digits). Alongside the sample space, we are given the label set, denoted \mathcal{Y} . This set contains the various classes to which the data points may be assigned (e.g., $0, 1, \dots, 9$ in the handwritten digits example). A distribution \mathcal{D} is defined over $\mathcal{X} \times \mathcal{Y}$, and dictates the probability to sample a data point $x \in \mathcal{X}$ along with a label $y \in \mathcal{Y}$. In our discussion we will

restrict ourselves to the Euclidean case, namely $\mathcal{X} = \mathbb{R}^d$. Unless stated otherwise, we assume a binary setting, in which $\mathcal{Y} = \{+1, -1\}$.

2. **Hypothesis class.** In the learning process, one considers candidate hypotheses taken out of the class \mathcal{H} . This class consists of functions from \mathcal{X} to \mathcal{Y} . Its contents reflect some kind of prior data about the problem at hand. A well known example is the class of half-spaces, defined as

$$\mathcal{H}_{half-space} = \{\phi_{\mathbf{w}}(\mathbf{x}) = \text{sgn}(\mathbf{w}^T \mathbf{x}) \mid \phi_{\mathbf{w}} : \mathbb{R}^d \rightarrow \{+1, -1\}, \mathbf{w} \in \mathbb{R}^d\} \quad (1.1)$$

3. **Loss measure.** The means to measure the performance of a specific instance $h \in \mathcal{H}$ is the loss function, $\ell : \mathcal{X} \times \mathcal{Y} \times \mathcal{H} \rightarrow \mathbb{R}_+$. The most intuitive loss function in the binary case is the zero-one loss, defined by

$$\ell_{0-1}(\mathbf{x}, y; h) = \mathbb{1}_{[h(\mathbf{x}) \neq y]} \quad (1.2)$$

The learning task is to find the classifier $h^* \in \mathcal{H}$ which is optimal, in the sense that it minimizes the *actual risk*, defined as

$$\text{err}(h) = \mathbb{E}_{(\mathbf{x}, y) \sim \mathcal{D}} \ell(\mathbf{x}, y; h) \quad (1.3)$$

Most of the times, however, it is the case that \mathcal{D} is unknown. Even in the rare cases in which it is known, it is not always possible to optimize the expectation over it. The learner is thus given a *training set* $\mathcal{S} \subseteq \mathcal{X} \times \mathcal{Y}$ of i.i.d. samples. The learning task in that case is to minimize the *empirical risk*, defined as

$$\hat{\text{err}}(h) = \frac{1}{M} \sum_{m=1}^M \ell(\mathbf{x}^m, y^m; h) \quad (1.4)$$

where $\mathcal{S} = \{(\mathbf{x}^m, y^m)\}_{m=1}^M$. This technique is called *empirical risk minimization* (ERM). It is important to keep in mind that although the technical tool is ERM, the objective is always to have the actual risk as low as possible.

Sometimes, however, this is not the case. That is, in spite of the fact that the learned decision rule is capable of classifying the training data, it fails to do so on fresh test data. In this case we say that the *generalization error* is high, although the training error is low. The reason for such a failure is most often *overfitting*. In this situation, the learned classifier fits the training data very well, but misses the general rule behind the data. In the PAC model, overfitting is explained by a too rich hypothesis class. If the learner can choose a model that fits perfectly the training data - it will do so, ignoring the fact that the chosen model will possibly not be able to explain new data. Say for example that the hypothesis class consists of all the functions from the sample space to the labels space. A naïve learner might choose a classifier that handles all the training points well, whereas any unknown sample is classified as $+1$. This selection might obviously have erroneous results. In the

spirit of this idea, the PAC theory bounds the difference between the empirical and the actual risk using a combinatorial measure of the hypothesis class complexity, named *VC dimension*. For a detailed review see Vapnik [1995].

A common solution for this problem is to add a regularization term to the objective of the minimization problem. Usually, a norm of the classifier is taken as a regularization term. From the statistical learning theory's point of view, the regularization restricts the complexity of the model, and by that controls the difference between the training and testing error (Smola et al. [1998]; Evgeniou et al. [2000]; Bartlett et al. [2002]). The idea of minimizing the complexity of the model is not unique to the statistical theory, and may be traced back to the Ocaam's razor principle: the simplest hypothesis that explains the phenomenon is likely to be the correct one. Another way to understand the regularization term, is as a means to introduce prior knowledge.

3. Support Vector Machines

In support vector machine (SVM), the loss measure at hand is the hinge-loss

$$\ell_{\text{hinge}}(\mathbf{x}^m, y^m; \mathbf{w}) = [1 - y^m \mathbf{w}^T \mathbf{x}^m]_+$$

ℓ_{hinge} is a surrogate loss function, in the sense that it upper-bounds the zero-one loss. Furthermore, ℓ_{hinge} is convex, which makes it a far more convenient objective for numerical optimization than the zero-one loss. Note that the hinge loss introduces penalty when the classifier correctly predicts the label of a sample, but does so with too little margin, i.e. $\mathbf{w}^T \mathbf{x}^m < 1$. The penalty on a wrong classification is linear in the distance of the sample from the hyperplane.

As discussed, optimizing the sum of the losses solely may result in poor generalization performance. The SVM solution is to add an L_2 regularization term. The geometrical intuition behind this term is the following: The distance between the point \mathbf{x}^m and the hyperplane $\mathbf{w}^T \mathbf{x} = b$ is given by

$$\frac{|\mathbf{w}^T \mathbf{x}^m - b|}{\|\mathbf{w}\|} \quad (1.5)$$

One may scale \mathbf{w} and b in such a way that the point with the smallest margin (that is, the one closest to the hyperplane) will have $\|\mathbf{w}^T \mathbf{x}^m - b\| = 1$. In that case, the bilateral margin is $\frac{2}{\|\mathbf{w}\|}$ (see Figure 1.2). This geometrical intuition, along with the fact that the hinge loss punishes too little margin, motivates the name *Maximum Margin Classification* that was granted to SVM. Hence, the SVM optimization task is

$$\min_{\mathbf{w}, b} \frac{\lambda}{2} \|\mathbf{w}\|^2 + \sum_{m=1}^M [1 - y^m (\mathbf{w}^T \mathbf{x}^m - b)]_+ \quad (1.6)$$

The parameter λ controls the tradeoff between the training error and the margin of the classifier (cf. Section 5).

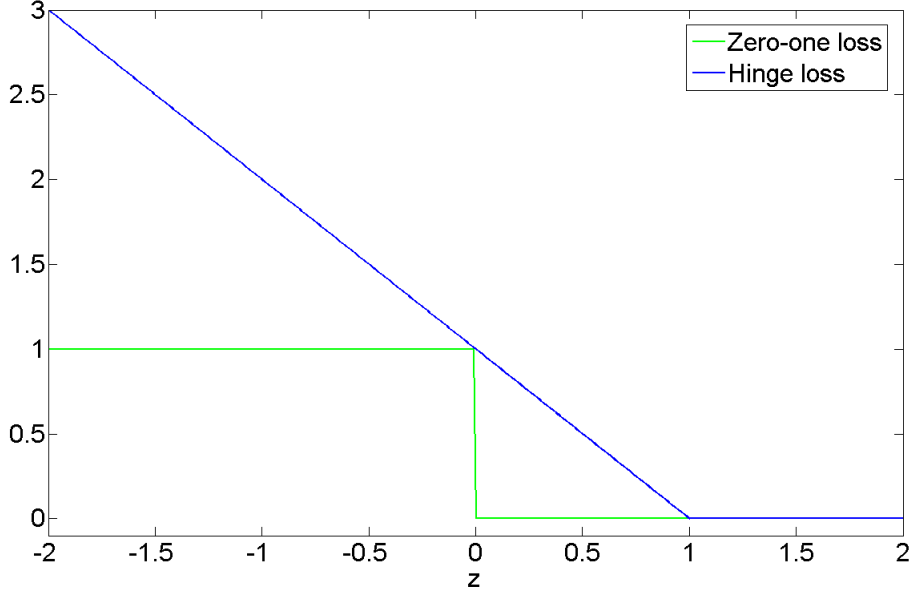


Figure 1.1: The hinge loss is a convex surrogate to the zero-one loss.

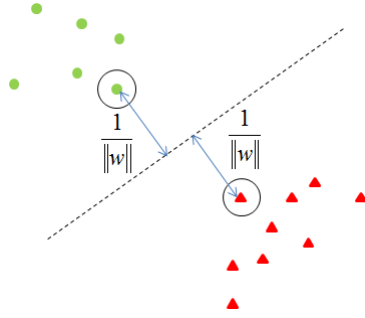


Figure 1.2: The bilateral margin is $\frac{2}{\|w\|}$. Thus, minimizing $\|w\|$ results in maximizing the margin.

4. Robustness

The objective of the learning is to be able to classify new data. Thus, being robust to perturbations of the data is usually a desirable property for a classifier. In some cases, the training data and the testing data are sampled from different processes, which are similar to some extent but are not identical (Bi and Zhang [2004]). This situation can happen also due to application specific issues, when new samples are sampled with reduced accuracy (for example, the training data may be collected with an expensive sensor, whereas cheaper sensors are deployed for actual use).

Even harder scenario is the one of learning in the presence of an adversary that may corrupt the training data, the testing data or both. The key step in order to formulate the robust learning task, is to model the action of the adversary, i.e., to define what is the family

of perturbations that he may apply on the data points. In the Robust-SVM model, the adversary may apply a bounded additive disturbance, by displacing a sample point within a ball around it (Shivaswamy et al. [2006]). This case is referred to as box-uncertainty. Globerson and Roweis [2006] assumed a different type of adversary. In their model, named FDROP, the adversary is allowed to delete a bounded number of features. This model results in more balanced classifiers, which are less likely to base their prediction only on a small subset of informative features.

Two issues usually repeat in robust learning formulations. The first one is the problem of the *adversarial choice*. Most of the times, the first step in the analysis of the model is characterizing the exact action of the adversary on a specific data sample, given specific model parameters. The Robust-SVM adversary will choose to displace the point perpendicularly to the separating hyperplane. FDROP's adversary will delete the most informative features, i.e. those that have the maximal contribution to the dot product between the weights vector and the data point. The second issue is the restriction on the adversary's action. Regardless of the actual type of perturbation that the adversary uses, one needs to bound the extent to which it is applied. If no constraint is specified, the adversary will choose his action in such a way that the signal to noise ratio (SNR) will vanish, and the data will no longer carry any information. In the Robust-SVM formulation, the adversary is constrained to perform a displacement within a bounded ball. In the case of FDROP, no more than a pre-defined number of features may be deleted.

Note that robust formulations are closely related to the notion of *consistency*. A classifier is said to be consistent, if close enough data points are predicted to have the same label. Different adversarial models befit different notions of distance. For example, the box-uncertainty model is related to the Euclidean metric and feature deletion suits the hamming distance.

It should be mentioned that robustness has quite a few meanings in the literature of statistics and machine learning. In this work, we use robustness in the sense of *robust optimization* (RO), i.e. minimizing the worst-case loss under given circumstances.

5. Between robustness and regularization

The fact the robustness is related to regularization and generalization is not too surprising. Indeed, first equivalence results have been established for learning problems other than classification more than a decade ago (Ghaoui and Le Bret [1997]; Xu et al. [2008]; Bishop [1994]). Recently, Xu et al. [2009] have proven the fact that the regularization employed by SVM is equivalent to a robust formulation. Specifically, they have shown that the following two formulations are equivalent

$$\min_{\mathbf{w}, b} \lambda \|\mathbf{w}\| + \sum_{m=1}^M [1 - y^m(\mathbf{w}^T \mathbf{x}^m - b)]_+$$

$$\min_{\mathbf{w}, b} \max_{\sum_m \|\delta_m\|^* \leq \lambda} \sum_{m=1}^M [1 - y^m(\mathbf{w}^T(\mathbf{x}^m - \delta_m) - b)]_+$$

where $\|\cdot\|_*$ is the dual-norm. This equivalence has a strong geometric interpretation, and sheds a new light on the function of the tuning parameter of SVM. Using the notion of robustness, a consistency result for SVM was given, without the use of VC or stability arguments. The novelty of that work stems from the fact that most previous works on robust classification were not aimed at relating robustness to regularization. Rather, the models were based on an already regularized SVM formulation, in which the loss measure was effectively modified.

6. Our contribution

In this work we adopt the idea of using robustness as a means to achieve generalization. We present a new robust-learning setup in which each data point is altered by a stochastic cloud centered on it. The loss is then evaluated as the expectation of an underlying loss on the cloud. The parameters of this cloud's distribution are chosen in an adversarial fashion. We analyze the case in which the adversary is restricted to choose a Gaussian cloud with a trace-bounded covariance matrix. Then we show that this formulation culminates in a smooth upper-approximation of the hinge loss, which gets tighter as the cloud around each data sample shrinks. This loss function can be shown to have a convex perspective structure. By deriving the dual problem, we are able to demonstrate a method of generating new smooth loss functions. Our algorithmic approach is to directly solve the primal problem. We show that this yields a learning algorithm which generalizes as well as SVM on synthetic as well as real data. Generalizations to the non-linear and multiclass cases are given.

7. Related work

Other works have incorporated a noise model into the learning setup. For example, baptiste Pothin and Richard [2008] have warped the data points with ellipsoids. Pozdnoukhov et al. [2005] have shown how to train classifiers for distributions. Similar to what we do in this work, they use tails of distributions in their derivation. Their work, however, treated each data class as a distribution, whereas in this work we attach a noise distribution for each data point separately. Bhattacharyya et al. [2004b]; Shivaswamy et al. [2006] have employed second order cone programming (SOCP) methods in order to handle the uncertainty in the data. Bhattacharyya et al. [2004a] have assumed stochastic clouds instead of discrete points, as we do, but they did not try to minimize the expectation of the loss function over the cloud. Instead, their idea was to incorporate the idea with the soft margin framework. Bi and Zhang [2004] have tried to learn a better classifier by presenting the learning algorithm 'more reasonable' samples. We elaborate on this model in Appendix A.

Smooth loss function were studied by Zhang et al. [2003]; Chapelle [2007]. Analysis of methods for Solving SVM and SVM-like problems using the primal formulation was done by Shalev-Shwartz et al. [2007a]; Chapelle [2007].

The rest of this document is organized as follows: in Chapter 2 we present our framework

formally, derive the explicit form of the smooth loss function and devise an algorithm that finds the optimal classifier. In Chapter 3 we derive a dual formulation for the problem, and point out that our model may be generalized for other loss functions. In Chapter 4 we apply the kernel trick and devise a method for training non-linear classifiers in the same cost as for the linear kernel. Chapter 5 contains a generalization of the binary algorithm for the multiclass case. At last, in Chapter 6 we discuss the contributions presented in this work and mention possible directions for future work. In Appendix A we discuss a far more basic version of resistance to noise. The results of the first section therein are not original and presented here only for the sake of logical order. The next section contains a simple generalization for the multiclass case. Appendix B gives the solution to the adversarial choice problem for an adversary that is restricted to spread the noise along the primary axes. At last, in Appendix C we explain why we find the usual multiclass hinge loss inapplicable in our framework.

Chapter 2

Gaussian Robust Classification

In this work we take the approach of robust optimization (RO). Accordingly, we present a min-max learning framework, in which the learner strives to minimize the loss, whereas the adversary tries to maximize it. The model that we introduce in this chapter has two layers of 'robustness'. Firstly, we use the min-max robustness, which lays in the foundations of RO. Secondly, we effectively enhance the training dataset by taking into consideration all the possible outputs of the adversarial perturbation. More concretely, we alter each training sample with a stochastic cloud. The shape of this cloud is chosen by the adversary from a pre-determined family of distributions. The spreading of the samples should be understood as adding noise, where different disturbances take place with different probability. The loss on each sample is finally computed as the expectation of an underlying loss on the respective cloud.

1. Problem Formulation

In this section we formally describe the model we investigate in the work. We take the hinge-loss as the underlying loss function, and build the learning framework on top of it. We then show that the new framework we introduce is equivalent to an unconstrained minimization of an effective loss function.

Recall that the hinge loss is defined

$$\ell_{hinge}(\mathbf{x}^m, y^m; \mathbf{w}) = [1 - y^m \mathbf{w}^T \mathbf{x}^m]_+ \quad (2.1)$$

We introduce the expected hinge loss

$$\ell_{hinge}^{\mathbb{E}}(\mathbf{x}^m, y^m; \mathbf{w}, \mathcal{D}) = \mathbb{E}_{\mathbf{n} \sim \mathcal{D}} [1 - y^m \mathbf{w}^T (\mathbf{x}^m + \mathbf{n})]_+ \quad (2.2)$$

where \mathcal{D} is a predefined noise distribution over the sample space. The optimization problem for learning a classifier w.r.t. the expected hinge loss is thus

$$\min_{\mathbf{w}} \sum_{m=1}^M \ell_{hinge}^{\mathbb{E}}(\mathbf{x}^m, y^m; \mathbf{w}, \mathcal{D}) \quad (2.3)$$

Granting an adversary the ability to choose the noise distribution, we end up with the following formulation

$$\min_{\mathbf{w}} \max_{\mathcal{D}_1 \times \mathcal{D}_2 \times \dots \times \mathcal{D}_M \in \mathcal{C}_1 \times \mathcal{C}_2 \times \dots \times \mathcal{C}_M} \sum_{m=1}^M \ell_{hinge}^{\mathbb{E}}(\mathbf{x}^m, y^m; \mathbf{w}, \mathcal{D}_m) \quad (2.4)$$

where \mathcal{C}_m is the set of allowed noise distributions for the m^{th} sample. In order for the adversarial optimization to be meaningful, all \mathcal{C}_m 's should have a 'bounded' nature. We now alter the order of maximization and summation, and write

$$\min_{\mathbf{w}} \sum_{m=1}^M \max_{\mathcal{D}_m \in \mathcal{C}_m} \ell_{hinge}^{\mathbb{E}}(\mathbf{x}^m, y^m; \mathbf{w}, \mathcal{D}_m) \quad (2.5)$$

At last, we observe that the optimization task at hand is nothing else than optimizing the effective loss function

$$\ell_{hinge}^{rob}(\mathbf{x}^m, y^m; \mathbf{w}, \mathcal{D}) = \max_{\mathcal{D} \in \mathcal{C}} \mathbb{E}_{\mathbf{n} \sim \mathcal{D}} [1 - y^m \mathbf{w}^T(\mathbf{x}^m + \mathbf{n})]_+ \quad (2.6)$$

We refer to $\ell_{hinge}^{rob}(\mathbf{x}^m, y^m; \mathbf{w}, \mathcal{D})$ as the expected robust hinge loss.

2. The adversarial Choice

Equation 2.5 presents the general noise-robust formulation. In the following, we will derive an explicit loss function for a specific collection of noise distributions. We focus on the case in which the adversary is constrained to spread a Gaussian noise, having a trace bounded covariance-matrix. The motivation behind this constraint is physical. When a noise is modeled with a distribution, its covariance is considered as its power. Thus, by constraining the sum of the eigenvalues of the covariance matrix we bound the power that the adversary can spread. The Gaussian noise is the worst case noise, in the sense that amongst all distributions with a certain power bound it has the maximal entropy.

Using the notations of the previous section, we specify the restriction on the adversary as

$$\mathcal{C} = \mathcal{C}_{\beta} = \{\mathcal{D} \sim \mathcal{N}(\mathbf{0}, \Sigma) | \Sigma \in \Lambda_{\beta}\}$$

where $\Lambda_{\beta} = \{\Sigma \in \text{PSD} | \text{Tr}(\Sigma) \leq \beta\}$, i.e. Gaussian distributions having the zero vector as mean and a covariance matrix with a bounded sum of eigenvalues.

In the next couple of sections we will characterize the adversarial choice of the covariance matrix and derive an explicit loss function.

2.1 The structure of the loss function

The following paragraphs are rather technical. For later use, we explicitly perform the integration of the robust hinge loss function. We then prove a monotony property of the integrated loss. This property will help us analyze the nature of the adversarial choice in

our case. The key observation throughout the derivation is that the multivariate expectation can be transformed to a univariate problem.

We plug the notations that were introduced above into Equation 2.6 and get:

$$\ell_{hinge}^{rob}(\mathbf{x}^m, y^m; \mathbf{w}, \Sigma) = \max_{\Sigma \in \Lambda_\beta} c|\Sigma|^{-\frac{1}{2}} \int e^{-\frac{1}{2}\mathbf{n}^T \Sigma^{-1} \mathbf{n}} [1 - y^m \mathbf{w}^T (\mathbf{x}^m + \mathbf{n})]_+ d\mathbf{n} \quad (2.7)$$

where $c = (2\pi)^{-d/2}$ is the normalization constant. This is equivalent to:

$$\ell_{hinge}^{rob}(\mathbf{x}^m, y^m; \mathbf{w}, \Sigma) = \max_{\Sigma \in \Lambda_\beta} c|\Sigma|^{-\frac{1}{2}} \int e^{-\frac{1}{2}\mathbf{n}^T \Sigma^{-1} \mathbf{n}} [1 - y^m \mathbf{w}^T \mathbf{x}^m - y^m \mathbf{w}^T \mathbf{n}]_+ d\mathbf{n} \quad (2.8)$$

As a first step in the analysis of the expected robust hinge loss, we shall handle the quantity

$$Q \stackrel{\text{def}}{=} c|\Sigma|^{-\frac{1}{2}} \int e^{-\frac{1}{2}\mathbf{n}^T \Sigma^{-1} \mathbf{n}} [1 - y^m \mathbf{w}^T \mathbf{x}^m - y^m \mathbf{w}^T \mathbf{n}]_+ d\mathbf{n} \quad (2.9)$$

Note that the above only depends on \mathbf{n} via products of the form $\mathbf{w}^T \mathbf{n}$. Therefore, we define a new scalar variable $u = y^m \mathbf{w}^T \mathbf{n}$. Equation 2.9 can now be viewed as the expected value of $g(u) = [1 - y^m \mathbf{w}^T \mathbf{x}^m - u]_+$. The moments of u are

$$\begin{aligned} \mathbb{E}u &= \mathbb{E}y^m \mathbf{w}^T \mathbf{n} = \\ &= y^m \mathbf{w}^T \mathbb{E}\mathbf{n} = 0 \end{aligned}$$

and

$$\begin{aligned} \text{Var}[u] &= \text{Var}[y^m \mathbf{w}^T \mathbf{n}] = \\ &= y^m \mathbf{w}^T \text{Var}[\mathbf{n}] y^m \mathbf{w} = \\ &= (y^m)^2 \mathbf{w}^T \Sigma \mathbf{w} = \\ &= \mathbf{w}^T \Sigma \mathbf{w} \end{aligned}$$

Thus we get

$$Q = \frac{1}{\sqrt{2\pi \mathbf{w}^T \Sigma \mathbf{w}}} \int e^{-\frac{u^2}{2\mathbf{w}^T \Sigma \mathbf{w}}} [1 - y^m \mathbf{w}^T \mathbf{x}^m - u]_+ du \quad (2.10)$$

Define $\text{erf}(t) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^t \exp(-\frac{1}{2}z^2) dz$. In addition, denote $\sigma^2(\mathbf{w}, \Sigma) = \mathbf{w}^T \Sigma \mathbf{w}$. The following proposition holds.

Proposition 1 $Q = (1 - y^m \mathbf{w}^T \mathbf{x}^m) \text{erf}\left(\frac{1 - y^m \mathbf{w}^T \mathbf{x}^m}{\sqrt{\sigma^2(\mathbf{w}, \Sigma)}}\right) + \sqrt{\frac{\sigma^2(\mathbf{w}, \Sigma)}{2\pi}} \exp\left(-\frac{(1 - y^m \mathbf{w}^T \mathbf{x}^m)^2}{2\sigma^2(\mathbf{w}, \Sigma)}\right)$

Proof: We conduct a direct computation:

$$\begin{aligned}
Q &= \frac{1}{\sqrt{2\pi\sigma^2(\mathbf{w}, \Sigma)}} \int e^{-\frac{u^2}{2\sigma^2(\mathbf{w}, \Sigma)}} [1 - y^m \mathbf{w}^T \mathbf{x}^m - u]_+ du = \\
&= \frac{1}{\sqrt{2\pi\sigma^2(\mathbf{w}, \Sigma)}} \int_{-\infty}^{1-y^m \mathbf{w}^T \mathbf{x}^m} e^{-\frac{u^2}{2\sigma^2(\mathbf{w}, \Sigma)}} (1 - y^m \mathbf{w}^T \mathbf{x}^m - u) du = \\
&= \frac{1}{\sqrt{2\pi\sigma^2(\mathbf{w}, \Sigma)}} \left((1 - y^m \mathbf{w}^T \mathbf{x}^m) \int_{-\infty}^{1-y^m \mathbf{w}^T \mathbf{x}^m} e^{-\frac{u^2}{2\sigma^2(\mathbf{w}, \Sigma)}} du \right. \\
&\quad \left. - \int_{-\infty}^{1-y^m \mathbf{w}^T \mathbf{x}^m} u e^{-\frac{u^2}{2\sigma^2(\mathbf{w}, \Sigma)}} du \right) = \\
&= (1 - y^m \mathbf{w}^T \mathbf{x}^m) \operatorname{erf} \left(\frac{1 - y^m \mathbf{w}^T \mathbf{x}^m}{\sqrt{\sigma^2(\mathbf{w}, \Sigma)}} \right) \\
&\quad - \frac{1}{\sqrt{2\pi\sigma^2(\mathbf{w}, \Sigma)}} \int_{-\infty}^{1-y^m \mathbf{w}^T \mathbf{x}^m} u e^{-\frac{u^2}{2\sigma^2(\mathbf{w}, \Sigma)}} du
\end{aligned}$$

By using the variable substitution theorem and observing that the remaining integrand is an odd function (thus the identity $\int_{-\infty}^t \text{odd} = \int_{-\infty}^{-t} \text{odd}$ holds), we conclude that

$$Q = (1 - y^m \mathbf{w}^T \mathbf{x}^m) \operatorname{erf} \left(\frac{1 - y^m \mathbf{w}^T \mathbf{x}^m}{\sqrt{\sigma^2(\mathbf{w}, \Sigma)}} \right) + \sqrt{\frac{\sigma^2(\mathbf{w}, \Sigma)}{2\pi}} \exp \left(-\frac{(1 - y^m \mathbf{w}^T \mathbf{x}^m)^2}{2\sigma^2(\mathbf{w}, \Sigma)} \right) \quad (2.11)$$

■

Let us establish the following simple property of Q .

Lemma 2 Q is monotone-increasing in σ^2 .

Proof: The fundamental theorem of calculus yields that

$$\frac{d}{dt} \operatorname{erf}(t) = \frac{1}{\sqrt{2\pi}} \exp \left(-\frac{t^2}{2} \right) \quad (2.12)$$

Using the chain rule we compute

$$\frac{dQ}{d\sigma^2} = \frac{1}{2\sqrt{2\pi}\sqrt{\sigma^2}} \exp \left(-\frac{(1 - y^m \mathbf{w}^T \mathbf{x}^m)^2}{2\sigma^2(\mathbf{w}, \Sigma)} \right) \quad (2.13)$$

It is evident that for all $\sigma^2 \geq 0$

$$\frac{dQ}{d\sigma^2} \geq 0 \quad (2.14)$$

i.e. Q is monotone-increasing in σ^2 .

■

2.2 The optimal covariance matrix subject to a trace constraint

We will now focus on finding the optimal adversary, i.e., performing the maximization of Equation 2.8 over the range of allowed covariance matrices. The next theorem specifies which covariance matrix attains the worst-case loss. In our terminology, refer to this result as the *adversarial choice*.

Theorem 2.1: The optimal Σ in Equation 2.8 is given by $\Sigma^* = \beta \frac{\mathbf{w}\mathbf{w}^T}{\|\mathbf{w}\|^2}$ where the optimization is done over $\Sigma \in \Lambda_\beta$.

Before actually proving the theorem, we will give some geometric intuition. The idea behind the expected loss is to replace the original sample point with a Gaussian cloud centered at the original point (Figure 2.1a). Consider an arbitrary displacement $\hat{\mathbf{x}}^m = \mathbf{x}^m + \mathbf{n}$. For fixed \mathbf{w} , \mathbf{n} can be written as $\mathbf{n} = \mathbf{n}_\parallel + \mathbf{n}_\perp$. The relevant quantity is $\mathbf{w}^T \hat{\mathbf{x}}^m = \mathbf{w}^T \mathbf{x}^m + \mathbf{w}^T \mathbf{n}_\parallel$, that is, the orthogonal component does not have any effect. Accordingly, it makes sense that the optimal noise direction is orthogonal to the separating hyperplane, i.e. parallel to the vector \mathbf{w} (see Figure 2.1b).

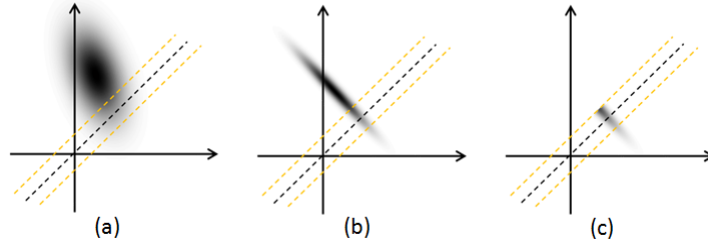


Figure 2.1: (a) Replacing the sample point with a Gaussian cloud. (b) The optimal noise direction is orthogonal to the separating hyperplane. (c) The expected robust hinge loss only considers the tail of the distribution, i.e. the points that suffer a margin error.

The proof of Theorem 2.1 applies simple algebraic results to establish this result rigorously.

Proof: Plugging Proposition 1 into Equation 2.8 we get

$$\begin{aligned} \ell_{hinge}^{rob}(\mathbf{x}^m, y^m; \mathbf{w}, \Sigma) &= \max_{\Sigma \in \Lambda_\beta} \left[(1 - y^m \mathbf{w}^T \mathbf{x}^m) \operatorname{erf} \left(\frac{1 - y^m \mathbf{w}^T \mathbf{x}^m}{\sqrt{\sigma^2(\mathbf{w}, \Sigma)}} \right) \right. \\ &\quad \left. + \sqrt{\frac{\sigma^2(\mathbf{w}, \Sigma)}{2\pi}} \exp \left(-\frac{(1 - y^m \mathbf{w}^T \mathbf{x}^m)^2}{2\sigma^2(\mathbf{w}, \Sigma)} \right) \right] \end{aligned}$$

The above depends on Σ only via $\sigma^2(\mathbf{w}, \Sigma)$. According to Lemma 2, the objective is monotone increasing in σ^2 . Therefore, the adversary would like to choose Σ so that

$\sigma^2(\mathbf{w}, \Sigma)$ is maximized. By applying the Cauchy-Schwartz inequality, we conclude that the maximum value of $\sigma^2(\mathbf{w}, \Sigma)$ is $\lambda_{\max}(\Sigma)\|\mathbf{w}\|^2$. For all $\Sigma \in \Lambda\beta$ it holds that $Tr(\Sigma) \leq \beta$. Since all of the eigenvalues are positive, it holds that $\lambda_{\max} \leq \beta$ as well. Consider the candidate solution $\Sigma_0 = \beta \frac{\mathbf{w}\mathbf{w}^T}{\|\mathbf{w}\|^2}$. Since $\sigma^2(\mathbf{w}, \Sigma_0) = \beta\|\mathbf{w}\|^2$, this selection attains the maximum. Note that this covariance matrix reflects the fact that the adversarial choice would be to spread the noise parallel to the separator. ■

3. A smooth loss function

In the previous sections we have done the technical computations needed in order to derive the robust hinge loss explicitly, and found the optimal covariance matrix. In the following we will put these results together, and present an explicit formulation of the loss function resulting from our model. In addition, it is shown that our robust loss can be represented as a perspective of a scalar smooth approximation of the hinge loss. By analyzing this function we are able to gain a better understanding of ℓ_{hinge}^{rob} . We conclude this section by showing that our loss function is a smooth convex upper-approximation of the hinge-loss. When the 'diameter' of the noise cloud is shrunk, ℓ_{hinge}^{rob} coincides with the hinge-loss.

We devote a notation for the result of Proposition 1

$$\begin{aligned} L(\mathbf{x}^m, y^m; \mathbf{w}, \sigma^2) &= (1 - y^m \mathbf{w}^T \mathbf{x}^m) \operatorname{erf}\left(\frac{1 - y^m \mathbf{w}^T \mathbf{x}^m}{\sigma}\right) \\ &+ \frac{\sigma}{\sqrt{2\pi}} \exp\left(-\frac{(1 - y^m \mathbf{w}^T \mathbf{x}^m)^2}{2\sigma^2}\right) \end{aligned}$$

By combining the above equation with the result of Theorem 2.1 we conclude

$$\begin{aligned} \ell_{hinge}^{rob}(\mathbf{x}^m, y^m; \mathbf{w}, \beta) &= (1 - y^m \mathbf{w}^T \mathbf{x}^m) \operatorname{erf}\left(\frac{1 - y^m \mathbf{w}^T \mathbf{x}^m}{\sqrt{\beta}\|\mathbf{w}\|}\right) \\ &+ \frac{\sqrt{\beta}\|\mathbf{w}\|}{\sqrt{2\pi}} \exp\left(-\frac{(1 - y^m \mathbf{w}^T \mathbf{x}^m)^2}{2\beta\|\mathbf{w}\|^2}\right) \end{aligned}$$

β has the meaning of statistical variance, and therefore in the following we will replace it with σ^2 (not to be confused with $\sigma^2(\mathbf{w}, \Sigma)$). In order to understand the nature of the loss function we have defined, it is suggestive to define

$$f(z) = z \operatorname{erf}(z) + \frac{1}{\sqrt{2\pi}} e^{-\frac{z^2}{2}} \quad (2.15)$$

Using f , the robust expected hinge loss can be written as

$$\ell_{hinge}^{rob}(\mathbf{x}^m, y^m; \mathbf{w}, \sigma^2) = \sigma\|\mathbf{w}\| f\left(\frac{1 - y^m \mathbf{w}^T \mathbf{x}^m}{\sigma\|\mathbf{w}\|}\right) \quad (2.16)$$

A direct computation shows that

$$\frac{df}{dz} = \text{erf}(z) + \frac{z}{\sqrt{2\pi}}e^{-\frac{z^2}{2}} - \frac{z}{\sqrt{2\pi}}e^{-\frac{z^2}{2}} = \text{erf}(z) \quad (2.17)$$

We are now ready to prove a simple yet fundamental property of f .

Theorem 3.1: f is a smooth strictly-convex upper-approximation of the hinge loss.

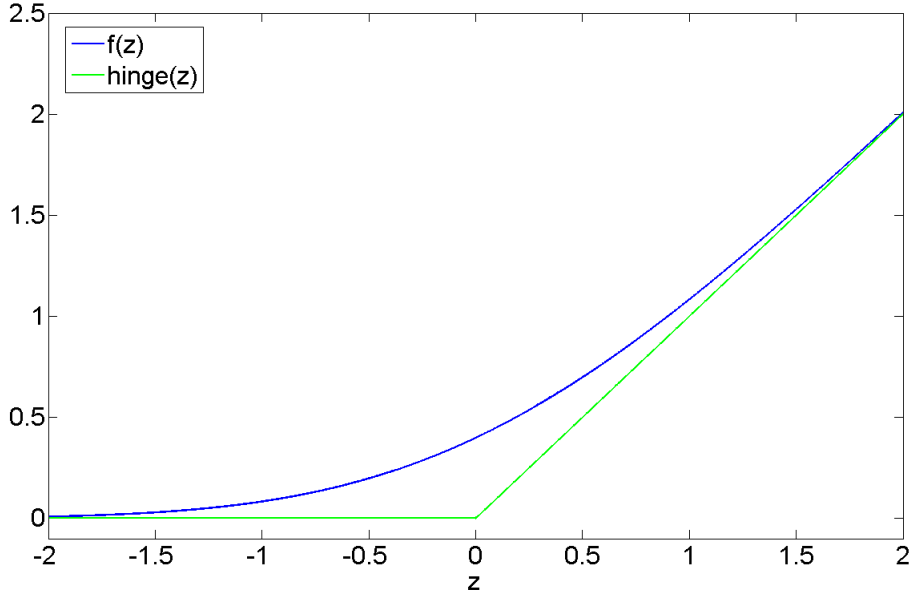


Figure 2.2: The function f is a smooth approximation of the hinge loss

Proof: Denote the hinge loss $h(z) = [z]_+$. We must show that

1. f is strictly-convex
2. $f(z) \geq h(z)$
3. $\lim_{z \rightarrow -\infty} f(z) - h(z) = 0$
4. $\lim_{z \rightarrow \infty} f(z) - h(z) = 0$

Differentiating Equation 2.17 once again, we get

$$\frac{d^2 f}{dz^2} = \frac{1}{\sqrt{2\pi}}e^{-\frac{z^2}{2}} \quad (2.18)$$

which is clearly positive for all z . Thus, f is strictly-convex.

For the upper bound property, notice that $f(z) \geq 0$ for all $z \in \mathbb{R}$. Hence, for $z < 0$ we

simply have $f(z) > 0 = h(z)$. For the complementary case, denote the difference function over the positives $\delta(z) = f(z) - h(z) = z\text{erf}(z) + \frac{1}{\sqrt{2\pi}}e^{-\frac{z^2}{2}} - z$. Using Equation 2.17 we obtain

$$\frac{dh}{dz} = \text{erf}(z) - 1 \quad (2.19)$$

It can be easily seen that $\frac{dh}{dz} < 0$, i.e. h is monotone decreasing. Observe that $\delta(0) = \frac{1}{\sqrt{2\pi}}$ and $\lim_{z \rightarrow \infty} \delta(z) = 0$. Since all the functions involved are continuous, we conclude that for $z \geq 0$ it holds that $h(z) \leq f(z) \leq h(z) + \frac{1}{\sqrt{2\pi}}$. Altogether, we have established the upper bound property.

For the asymptote at $z \rightarrow -\infty$, observe that from l'Hopital

$$\lim_{z \rightarrow -\infty} z\text{erf}(z) = \lim_{z \rightarrow -\infty} \sqrt{2\pi}e^{-\frac{z^2}{2}} = 0$$

Since the exponent in the right summand of f decays as well, we have that at $z \rightarrow -\infty$ both $f(z)$ and $h(z)$ coincide on $z = 0$.

For the asymptote at $z \rightarrow \infty$, we must show that f asymptotically coincide with the linear function z . To this end, let us write $f(z) - z = z(\text{erf}(z) - 1) + \frac{1}{\sqrt{2\pi}}e^{-\frac{z^2}{2}}$. Applying l'Hopital rule along with the asymptotic behavior of the exponent, we deduce that $\lim_{z \rightarrow \infty} f(z) - z = 0$, as desired. ■

Next, we will analyze the relation between f and $\ell_{\text{hinge}}^{\text{rob}}$.

Definition 3 *Perspective of a function* (from Boyd and Vandenberghe [2004] 3.2.6).

If $f : \mathbb{R}^n \rightarrow \mathbb{R}$, then the perspective of f is the function $g : \mathbb{R}^{n+1} \rightarrow \mathbb{R}$ defined by

$$g(t, x) = tf\left(\frac{x}{t}\right)$$

with domain

$$\text{dom}(g) = \left\{ (x, t) \mid \frac{x}{t} \in \text{dom}(f), t > 0 \right\}$$

■

The following lemma is useful. For a proof see Boyd and Vandenberghe [2004] 3.2.6, e.g.

Lemma 4 If f is convex (concave), then its perspective is convex (concave) as well.

Define the function

$$g(a, b) = af\left(\frac{b}{a}\right) \quad (2.20)$$

Lemma 4 implies that $g(\sigma\|\mathbf{w}\|, 1 - y^m\mathbf{w}^T\mathbf{x}^m)$ is jointly convex in both its arguments. In order to establish the strict-convexity of $\ell_{\text{hinge}}^{\text{rob}}$ in \mathbf{w} , we need a more powerful tool. Consider the following lemma (Boyd and Vandenberghe [2004] 3.2.4)

Lemma 5 Let $h : \mathbb{R}^k \rightarrow \mathbb{R}$, $g_i : \mathbb{R}^n \rightarrow \mathbb{R}$. Consider the function $f(x) = h(g(x)) = h(g_1(x), g_2(x), \dots, g_k(x))$. Then, f is convex if h is convex, h is nondecreasing in each argument, and g_i are convex.

This lemma can be easily generalized to the case of strictly-convex functions. The proof is identical to that of the original version, thus will be skipped.

We are now ready to prove the following theorem

Theorem 3.2: ℓ_{hinge}^{rob} is strictly-convex in \mathbf{w} .

Proof: From Lemma 4, g is convex. In addition, g is nondecreasing in each of its arguments. To see that, observe that

$$\begin{aligned} \frac{dg}{da} &= f\left(\frac{b}{a}\right) - \frac{b}{a} f'\left(\frac{b}{a}\right) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{1}{s} \frac{b^2}{a^2}\right) \\ \frac{dg}{db} &= f'\left(\frac{b}{a}\right) = \text{erf}\left(\frac{b}{a}\right) \end{aligned}$$

which are both strictly positive.

$\sigma\|\mathbf{w}\|$ and $(1 - y^m \mathbf{w}^T \mathbf{x}^m)$ are both convex in \mathbf{w} , thus we conclude by applying Lemma 5.

■

The next theorem explores some of the other properties of the loss function we have defined:

Theorem 3.3: ℓ_{hinge}^{rob} is an upper-approximation to the hinge loss. Furthermore, when $\sigma^2 \rightarrow 0$, the loss function ℓ_{hinge}^{rob} coincides with the hinge loss.

Proof: For the upper bound property, we apply Theorem 3.1:

$$\begin{aligned} \sigma\|\mathbf{w}\| f\left(\frac{1 - y^i \mathbf{w}^T \mathbf{x}^i}{\sigma\|\mathbf{w}\|}\right) &\geq \sigma\|\mathbf{w}\| h\left(\frac{1 - y^i \mathbf{w}^T \mathbf{x}^i}{\sigma\|\mathbf{w}\|}\right) \\ &= \sigma\|\mathbf{w}\| \left[\frac{(1 - y^i \mathbf{w}^T \mathbf{x}^i)}{\sigma\|\mathbf{w}\|} \right]_+ \\ &= [1 - y^i \mathbf{w}^T \mathbf{x}^i]_+ \end{aligned}$$

For the second part of the theorem, let us first observe that

$$\frac{\sigma}{\sqrt{2\pi}} \exp\left(-\frac{(1 - y^m \mathbf{w}^T \mathbf{x}^m)^2}{2\sigma^2}\right) \rightarrow 0 \quad (2.21)$$

as a multiplication of two vanishing factors at $\sigma \rightarrow 0$. We consider two cases:

1. $1 - y^m \mathbf{w}^T \mathbf{x}^m \geq 0$. Observe that

$$\operatorname{erf}\left(\frac{1 - y^m \mathbf{w}^T \mathbf{x}^m}{\sigma \|\mathbf{w}\|}\right) \rightarrow \operatorname{erf}(\infty) = 1$$

Thus, $\ell_{\text{hinge}}^{\text{rob}}(\mathbf{x}^m, y^m; \mathbf{w}, \sigma^2) \rightarrow 1 - y^m \mathbf{w}^T \mathbf{x}^m$.

2. $1 - y^m \mathbf{w}^T \mathbf{x}^m < 0$. In this case

$$\operatorname{erf}\left(\frac{1 - y^m \mathbf{w}^T \mathbf{x}^m}{\sigma \|\mathbf{w}\|}\right) \rightarrow \operatorname{erf}(-\infty) = 0$$

Thus, $\ell_{\text{hinge}}^{\text{rob}}(\mathbf{x}^m, y^m; \mathbf{w}, \sigma^2) \rightarrow 0$

Altogether, we have shown that when $\sigma \rightarrow 0$, $\ell_{\text{hinge}}^{\text{rob}}(\mathbf{x}^m, y^m; \mathbf{w}, \sigma^2) \rightarrow [1 - y^m \mathbf{w}^T \mathbf{x}^m]_+$.

■

Observe that at $\mathbf{w} = 0$ the loss function is not continuous. The discontinuity is removable, however, so this issue does not pose any problem.

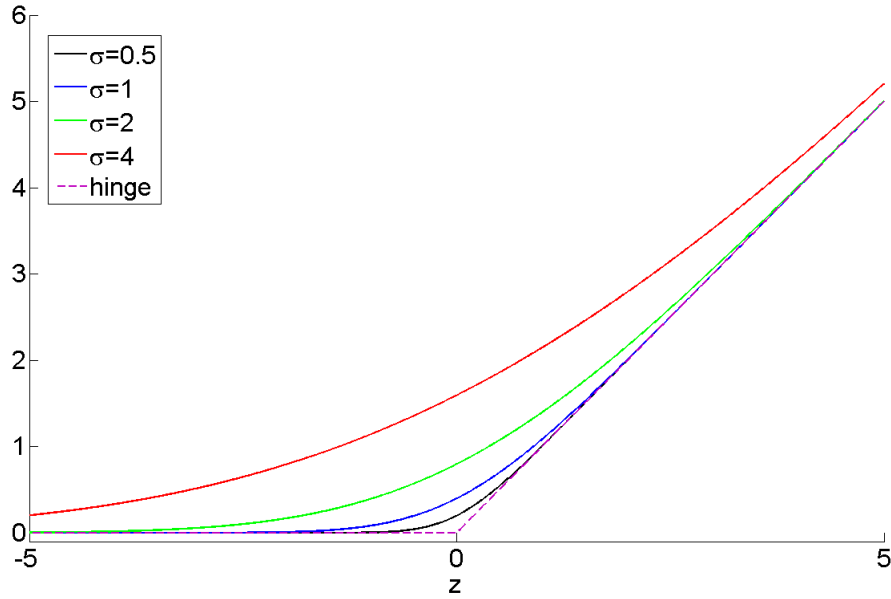


Figure 2.3: $\ell_{\text{hinge}}^{\text{rob}}$ is a convex upper-approximation to the hinge loss. As σ tends to 0, $\ell_{\text{hinge}}^{\text{rob}}$ tends to the hinge. In all of the graphs, the norm $\|\mathbf{w}\|$ was set to 1.

The norm of the classifier $\|\mathbf{w}\|$ always appears in a multiplication with σ . Thus, we observe that it has a similar function. Namely, it controls the tightness of the approximation of the smooth loss function to the hinge. Since σ is pre-determined, the optimal norm should reflect some kind of compensation. We thus conjecture that there exist a inverse ratio between σ and the optimal norm (cf. Chapter 4).

At last, it should be noted that this loss smooth function can be viewed as a multiplicative regularized loss.

4. GURU: a primal algorithm

We are now ready to devise an algorithm that solves our learning problem. In this section we describe a stochastic gradient descent (SGD) method that minimizes the strictly-convex loss function at hand. A convergence result for the algorithm stems from general properties of SGD that were studied extensively (see Shalev-Shwartz et al. [2007b]; Kivinen et al. [2003]; Zhang et al. [2003]; Nedic and Bertsekas [2000]; Bottou and Bousquet [2008], e.g.).

Plugging the robust hinge loss function we have derived (Equation 2.15) into the original optimization task (Equation 2.5), we get

$$\min_{\mathbf{w}} \sum_{m=1}^M (1 - y^m \mathbf{w}^T \mathbf{x}^m) \operatorname{erf} \left(\frac{1 - y^m \mathbf{w}^T \mathbf{x}^m}{\sigma \|\mathbf{w}\|} \right) + \frac{\sigma \|\mathbf{w}\|}{\sqrt{2\pi}} \exp \left(-\frac{(1 - y^m \mathbf{w}^T \mathbf{x}^m)^2}{2\sigma^2 \|\mathbf{w}\|^2} \right) \quad (2.22)$$

This formulation is a convex unconstrained minimization task. One very natural approach for solving this kind of task is using the family gradient descent methods. Denote the objective of the optimization as

$$G(\mathbf{w}) = \sum g_i(\mathbf{w}) \quad (2.23)$$

In batch gradient descent, in each step the algorithm updates

$$\mathbf{w} \leftarrow \mathbf{w} - \eta \nabla G(\mathbf{w}) \quad (2.24)$$

In stochastic gradient methods the gradient is approximated as the gradient of one of the summands. Thus, the algorithm first randomizes an index i , then updates

$$\mathbf{w} \leftarrow \mathbf{w} - \eta \nabla g_i(\mathbf{w}) \quad (2.25)$$

where η is the learning rate. The stochastic version suits settings of online learning, in which the learner is presented one training sample at a time. It has been suggested that using the stochastic version yields better generalization performance in learning tasks (Amari [1998]; Bottou and LeCun [2003]).

Our algorithm, named GURU (**GaUssian RobUst**), optimizes Equation 2.22 using an SGD procedure. (For a full treatment see, e.g. Boyd (ref).)

In order to derive the update formula, one should first calculate the gradient of the loss function. A straight forward computation yields

$$\nabla_{\mathbf{w}} \ell_{\text{hinge}}^{\text{rob}}(\mathbf{x}^i, y^i; \mathbf{w}, \sigma^2) = -y^i \mathbf{x}^i \operatorname{erf} \left(\frac{1 - y^i \mathbf{w}^T \mathbf{x}^i}{\sigma \|\mathbf{w}\|} \right) + \frac{\sigma \mathbf{w}}{\sqrt{2\pi} \|\mathbf{w}\|} \exp \left(-\frac{(1 - y^i \mathbf{w}^T \mathbf{x}^i)^2}{2\sigma^2 \|\mathbf{w}\|^2} \right) \quad (2.26)$$

Name	#Training samples	#Cross- validation samples	#Test samples	#features
Toy(a)	200	200	200	2
Toy(b)	200	200	200	2
Ionosp- here	100	100	152	34
diabetes	200	100	468	8
splice	500	400	635	60
1 vs. 2				
USPS	800	700	700	256
3 vs. 5				
USPS	800	700	700	256
5 vs. 8				
USPS	800	700	700	256
7 vs. 9				

Table 2.1: Description of the databased used in the binary case

We therefore suggest the following SGD procedure

Algorithm 1: GURU($\mathcal{S}, \eta_0, \epsilon$)

Data: Training set \mathcal{S} , learning rate η_0 , accuracy ϵ

Result: w

$w \leftarrow \mathbf{0};$

while $\Delta L \geq \epsilon$ **do**

$m \leftarrow \text{rand}(M);$
 $w \leftarrow w - \frac{\eta_0}{\sqrt{t}} \nabla_w \ell_{\text{hinge}}^{\text{rob}}(\mathbf{x}^m, y^m; w, \sigma^2);$

end

return $w;$

For convergence results see Nedic and Bertsekas [2000]. For a full treatment, see Bertsekas et al. [2003], chapter 8.

5. Experiments

In this section we present experimental results that demonstrate the fact that GURU generalizes as well as SVM. Experiments were carried out on two toy problems (see Figure 2.4 for a visualization), USPS handwritten digits classification (3 vs. 5, 5 vs. 8 and 7 vs. 9) and a couple of UCI databases (Frank and Asuncion [2010]). The sizes of the data sets are detailed in Table 2.1.

Name	GURU(%)	SVM(%)
Toy(a)	92.5	92.5
Toy(b)	92	92
Ionosp- here	82.24	79.61
diabetes	67.52	67.31
splice	93.39	92.44
1 vs. 2		
USPS	95.57	95.86
3 vs. 5		
USPS	97.71	98
5 vs. 8		
USPS	97.57	97.43
7 vs. 9		

Table 2.2: Summary of the results: GURU and SVM.

We have trianed GURU for σ values varying from 2^{-20} to 2^{20} , with exponential jumps. The learning rate was tuned empirically (values between 4^{-10} to 4^{10} were tested). SVM was trained and tested using the SVM-light package. λ values between 4^{-15} and 4^{15} were used. Note that in the SVM-light formulation, λ multiples the loss and not the regularization term. Thus, the qualitative relation between λ and σ is roughly $\sigma \sim \frac{1}{\lambda}$. Parameter selection was done based on the cross-validation set, and performance was evaluated for the optimal parameter on a testing set. The results are summarized in Table 2.2.

On the toy databases (a)-(b), the performance of GURU is identical to that of SVM. We have tested the learned classifiers' resistance to Noise, by adding uniformly distributed random noise to both cross-validation and test sets. The results are presented in Figure 2.5. Observe that the resistance of GURU slightly outperforms that of SVM. Nontheless, this result gives an experimental support to the theoretical result in Xu et al. [2009], where it was shown that the ordinary SVM formulation is equivalent to a robust formulation, in which the adversary is capable of displacing the data samples.

On the Ionosphere database, GURU significantly outperforms SVM. The samples of this database consist of radar reading. Thus, GURU's performance may be understood by the noisy nature of the samples. This finding supports the intuition that GURU perfoms well in noisy setups.

On USPS, the performance of GURU is pretty similar to that of SVM. Since the samples can be easily visualized as images, it is convenient to examine the adversarial action in this case. Consider Figure 2.6. The GURU adversary is symmetric, in the sense that it may move the samples either closer or further from the separating hyperplane. Hence, some digits look even more clear that the original ones, whereas others look as the opponent digit.

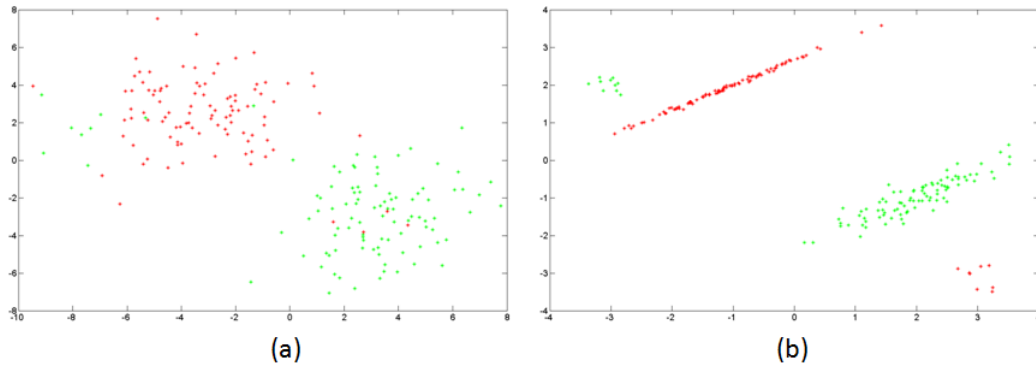


Figure 2.4: (a) Gaussian data. (b) Narrow Gaussian with outliers.

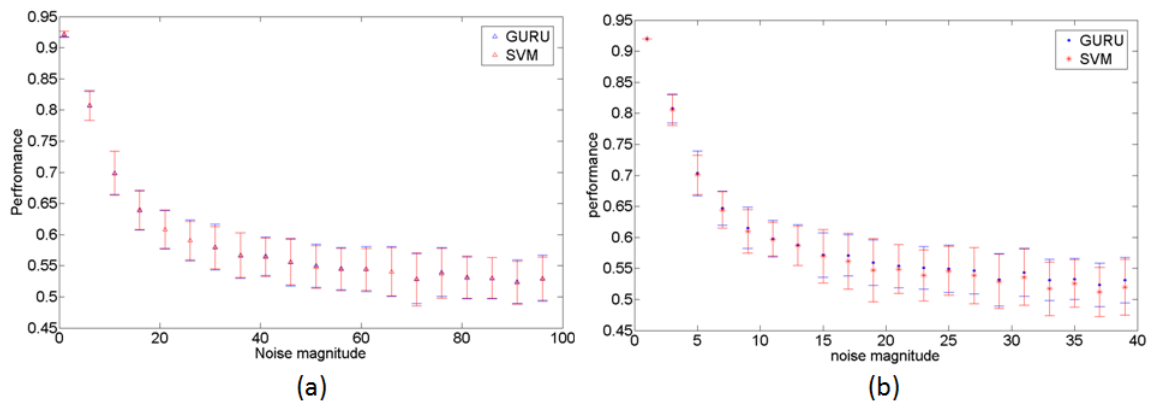


Figure 2.5: Classifiers' performance on a noised cross-validation and testing sets. The x -axis indicates the magnitude of the noise (noise distributes as $U(-x, x)$). The experiment was repeated 50 times. (a)-(b) represent the respective toy problems.

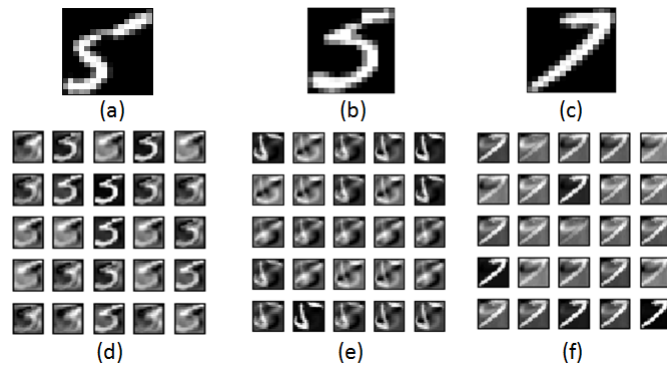


Figure 2.6: The GURU adversary adds noise perpendicularly to the separating hyperplane. Note that some samples are even more clear than the original, whereas others look like the opponent digit. A bunch of samples are a superposition of both. (a)-(c) are the original digits. (d)-(f) are 25 noisy samples.

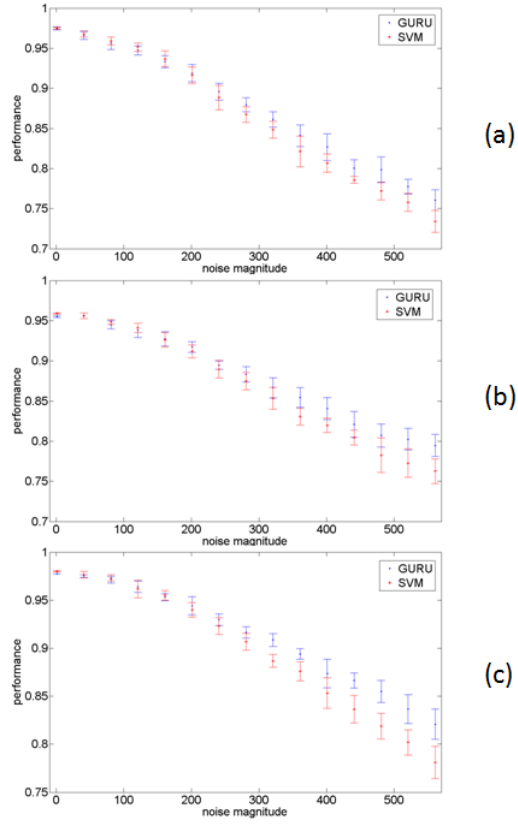


Figure 2.7: Classifiers' performance on a noised cross-validation and testing sets. The x -axis indicates the manitude of the noise (noise distributes as $U(-x, x)$). The experiment was repeated 10 times. (a) 3 vs 5, (b) 5 vs 8, (c) 7 vs 9.

In addition, on the USPS dataset, GURU has demonstrated a significantly better resistance to noise than SVM (see Figure 2.7).

Chapter 3

Dual formulation

In this chapter we derive a dual problem for the learning task at hand. We do not use the dual as a means to solve the primal problem, since the primal optimization works well. Rather, we use it to gain a better understanding of the problem. In the course of the derivation we use the notion of conjugate functions. We will show that the dual problem itself specifies the classifier up to a scaling factor. Thus, we devise a method to extract the norm using the available information. It is interesting to observe that throughout the derivation of the dual, the smooth function f plays a specific and distinguished role. Thus, the entire procedure may be applied as is for other smooth convex function, by only calculating their conjugate dual. We demonstrate this principle in Section 2, where we also discuss the relation between the primal loss and the dual formulation.

1. Mathematical Derivation

This section is rather technical, and goes through the derivation of the dual. We start with the perspective representation of ℓ_{hinge}^{rob} , and introduce couple of auxiliary variables. Using these variables, the Lagrangian takes a form that we are able to analyze. Theorem 1.1 encapsulates the effect of f , in such a manner that other loss functions can be plugged into the derivation rather easily.

The main result of this section is that the dual form of Equation 2.22 is

$$\begin{aligned} \max \quad & \sum_m \alpha_m \\ \text{s.t.} \quad & \left\| \sum_m \alpha_m y^m \mathbf{x}^m \right\| \leq \sigma \sum_m \frac{1}{\sqrt{2\pi}} \exp \left(-\frac{\text{erf}_{inv}^2(\alpha_m)}{2} \right) \\ & \alpha \geq 0 \end{aligned} \tag{3.1}$$

In the following paragraphs we will go through the details.

The optimization task Equation 2.22 may be written as

$$\min_{\mathbf{w}} \sigma \|\mathbf{w}\| \sum_m f \left(\frac{1 - y^m \mathbf{w}^T \mathbf{x}^m}{\sigma \|\mathbf{w}\|} \right) \tag{3.2}$$

We introduce the auxilliary variables z_m , and constrain them with $1 - y^m \mathbf{w}^T \mathbf{x} \leq z_m$. Note that $f(z)$ is monotone increasing in z . Thus, at optimality $z_m = 1 - y^m \mathbf{w}^T \mathbf{x}$. In addition, we introduce the variable r and constrain it with $\sigma \|\mathbf{w}\| \leq r$. At the optimum $r = \sigma \|\mathbf{w}\|$, since $r f(\frac{z}{r})$ is monotone increasing in r . Altogether we get the following optimization task

$$\begin{aligned} \min \quad & r \sum_m f\left(\frac{z_m}{r}\right) \\ \text{s.t.} \quad & \sigma \|\mathbf{w}\| \leq r \\ & 1 - y^m \mathbf{w}^T \mathbf{x}^m \leq z_m \\ & r \geq 0 \end{aligned} \tag{3.3}$$

where the optimization variables are $\mathbf{w}, z_1, \dots, z_n, r$.

The objective is convex according to Lemma 4, and the constraint on \mathbf{w} is a second order cone.

To find the dual, write the Lagrangian:

$$\mathcal{L}(\mathbf{w}, r, \mathbf{z}, \boldsymbol{\alpha}, \lambda) = r \sum_m f\left(\frac{z_m}{r}\right) + \lambda [\sigma \|\mathbf{w}\| - r] + \sum_m \alpha_m [1 - y^m \mathbf{w}^T \mathbf{x}^m - z_m] - \mu r$$

where $\lambda \alpha_m, \mu \geq 0$ are the Lagrange multipliers. For later convenience we add a set of variables r_m and force them all to equal r . So the new Lagrangian is:

$$\begin{aligned} \mathcal{L}(\mathbf{w}, r, \mathbf{z}, \boldsymbol{\alpha}, \lambda, \boldsymbol{\delta}) &= \sum_m r_m f\left(\frac{z_m}{r_m}\right) + \lambda [\sigma \|\mathbf{w}\| - r] \\ &+ \sum_m \alpha_m [1 - y^m \mathbf{w}^T \mathbf{x}^m - z_m] - \mu r - \sum_m \delta_m [r_m - r] \end{aligned}$$

where $\delta_m \geq 0$.

Recall that we have defined $g(a, b) = af\left(\frac{b}{a}\right)$. Using this notion we get the following task

$$\begin{aligned} \min_{\mathbf{w}, r, \mathbf{z}} \mathcal{L}(\mathbf{w}, r, \mathbf{z}, \boldsymbol{\alpha}, \lambda, \boldsymbol{\delta}) &= \min_{\mathbf{w}, r} \sum_m \min_{z_m, r_m} [g(r_m, z_m) - \alpha_m z_m - \delta_m r_m] \\ &+ \lambda [\sigma \|\mathbf{w}\| - r] + \sum_m \alpha_m [1 - y^m \mathbf{w}^T \mathbf{x}^m] - \mu r + r \sum_m \delta_m \\ &= \min_{\mathbf{w}, r} \sum_m g^*(\alpha_m; \delta_m) + \lambda [\sigma \|\mathbf{w}\| - r] \\ &+ \sum_m \alpha_m [1 - y^m \mathbf{w}^T \mathbf{x}^m] - \mu r + r \sum_m \delta_m \end{aligned}$$

where g^* is by definition the conjugate function of g (for details see Boyd and Vandenberghe [2004] 3.3, e.g). Deriving the Lagrangian w.r.t. \mathbf{w} gives:

$$\sigma \lambda \frac{\mathbf{w}}{\|\mathbf{w}\|} = \sum_m \alpha_m y^m \mathbf{x}^m \tag{3.4}$$

Taking the norm of both sides of the equation yields

$$\sigma\lambda(\boldsymbol{\alpha}) = \left\| \sum_m \alpha_m y^m \mathbf{x}^m \right\| \quad (3.5)$$

Substituting this back into the objective, the terms with \mathbf{w} cancel out and we have:

$$\min_r \sum_m g^*(\alpha_m; \delta_m) - r\lambda(\boldsymbol{\alpha}) - \mu r + r \sum_m \delta_m \quad (3.6)$$

This is linear in r , thus deriving w.r.t. r yields a constraint $\sum_m \delta_m = \lambda(\boldsymbol{\alpha}) + \mu$. Since $\mu \geq 0$, the equality constraint might be relaxed to $\sum_m \delta_m \geq \lambda(\boldsymbol{\alpha})$, and we end up with the following formulation

$$\begin{aligned} \max \quad & \sum_m \alpha_m + \sum_m g^*(\alpha_m; \delta_m) \\ \text{s.t.} \quad & \sum_m \delta_m \geq \lambda(\boldsymbol{\alpha}) \\ & \boldsymbol{\alpha} \geq 0 \end{aligned} \quad (3.7)$$

Or:

$$\begin{aligned} \max \quad & \sum_m \alpha_m + \sum_m g^*(\alpha_m; \delta_m) \\ \text{s.t.} \quad & \left\| \sum_m \alpha_m y^m \mathbf{x}^m \right\| \leq \sigma \sum_m \delta_m \\ & \boldsymbol{\alpha} \geq 0 \end{aligned} \quad (3.8)$$

The overall problem has a concave objective (since it's a conjugate dual of a convex function) and second order cone constraints. In what follows we work out the form of the conjugate dual g^* .

Denote by $f^*(\alpha)$ the conjugate function of f (it is concave). The next theorem specifies the conjugate g^* in terms of f^* :

Theorem 1.1: The conjugate dual of $g(a, b)$ is

$$g^*(\alpha, \delta) = \begin{cases} 0 & f^*(\alpha) \geq \delta \\ -\infty & \text{otherwise} \end{cases} \quad (3.9)$$

Proof: We must calculate

$$g^*(\alpha; \delta) = \min_{x, t} \left(t f\left(\frac{x}{t}\right) - \alpha x - \delta t \right) \quad (3.10)$$

To prove, we change from variables x, t to a variables $z = x/t, t$:

$$\min_{t \geq 0, z} t f(z) - \alpha z t - \delta t = \min_{t \geq 0, z} t(f(z) - \alpha z - \delta) \quad (3.11)$$

For the first case, assume that $f^*(\alpha) \geq \delta$, which implies that for all z :

$$f(z) - \alpha z \geq \delta \quad (3.12)$$

Then in Equation 3.11 the minimization is always of the product of $t \geq 0$ and some non-negative number. Hence it is always greater than zero, and zero can be attained at the limit $t \rightarrow 0$.

On the other hand if $f^*(\alpha) < \delta$, we will show that there exists a pair t, z that achieves a value $-\infty$: Since $f^*(\alpha) < \delta$ there exists a z for which

$$f(z) - \alpha z - \delta < 0 \quad (3.13)$$

If we take $t \rightarrow \infty$ and this z we get a value of $-\infty$. ■

In order to complete the derivation of the dual formulation, we should compute the conjugate dual f^* . The following lemma gives the desired result

Lemma 6 The conjugate dual of f is

$$f^*(\alpha) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{\text{erf}_{inv}^2(\alpha)}{2}\right)$$

Proof: Recall that:

$$f(z) = z\text{erf}(z) + \frac{1}{\sqrt{2\pi}} e^{-\frac{z^2}{2}} \quad (3.14)$$

and that its first derivative is

$$\frac{df}{dz} = \text{erf}(z)$$

(see Equation 2.17). By Theorem 3.1, f is convex, thus we compute f 's conjugate dual:

$$f^*(\alpha) = \min_z f(z) - \alpha z \quad (3.15)$$

The minimum satisfies:

$$\begin{aligned} f'(z) &= \alpha \\ \text{erf}(z) &= \alpha \\ z &= \text{erf}_{inv}(\alpha) \end{aligned}$$

where erf_{inv} is the inverse function of erf . We plug this equality into the objective and conclude

$$\begin{aligned} f^*(\alpha) &= f(\text{erf}_{inv}(\alpha)) - \alpha \text{erf}_{inv}(\alpha) \\ &= \text{erf}_{inv}(\alpha)\alpha + \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{\text{erf}_{inv}^2(\alpha)}{2}\right) - \alpha \text{erf}_{inv}(\alpha) \\ &= \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{\text{erf}_{inv}^2(\alpha)}{2}\right) \end{aligned}$$

It can be easily verified that f^* is concave, as expected from the theory. Note that from the derivation above it follows that $\alpha_M \leq 1$. ■

Taking the dual problem in Equation 3.8 and plugging in the conjugate duals derived above, we get:

$$\begin{aligned} \max \quad & \sum_m \alpha_m \\ \text{s.t.} \quad & \left\| \sum_m \alpha_m y^m \mathbf{x}^m \right\| \leq \sigma \sum_m \delta_m \\ & \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{\text{erf}_{inv}^2(\alpha_m)}{2}\right) \geq \delta_m \\ & \boldsymbol{\alpha} \geq 0 \end{aligned} \quad (3.16)$$

Consider the following problem:

$$\begin{aligned} \max \quad & \sum_m \alpha_m \\ \text{s.t.} \quad & \left\| \sum_m \alpha_m y^m \mathbf{x}^m \right\| \leq \sigma \sum_m \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{\text{erf}_{inv}^2(\alpha_m)}{2}\right) \\ & \boldsymbol{\alpha} \geq 0 \end{aligned} \quad (3.17)$$

The following proposition asserts that both of the formulations above are equivalent.

Proposition 7 *Equation 3.16 and Equation 3.17 are equivalent.*

Proof: Denote \mathcal{C}_1 the feasible region of Equation 3.16, and \mathcal{C}_2 the feasible region of Equation 3.17. Let $\boldsymbol{\alpha} \in \mathcal{C}_1$. Then trivially we have $\boldsymbol{\alpha} \in \mathcal{C}_2$. On the other hand, let $\boldsymbol{\alpha} \in \mathcal{C}_2$. Denote $\delta_m = \frac{1}{\sqrt{2\pi}} \exp(-\frac{1}{2}\text{erf}_{inv}^2(\alpha_m))$. It is easy to verify that this selection corresponds to a feasible point for Equation 3.16 (i.e. $\in \mathcal{C}_1$) with the same objective value. ■

As we have seen, the optimization problem we analyze in this work is a relative of the SVM problem. It is interesting to examine what happens when considering the duals. Consider the SVM formulation

$$\begin{aligned} \min_{\mathbf{w}} \quad & \frac{\lambda}{2} \|\mathbf{w}\|^2 + \sum_{m=1}^M \xi_m \\ \text{s.t.} \quad & \forall m \in \{1, 2, \dots, M\} : \xi_m \geq 1 - y^m \mathbf{w}^T \mathbf{x}^m \\ & \xi_m \geq 0 \end{aligned} \quad (3.18)$$

Its dual is

$$\begin{aligned} \min_{\boldsymbol{\alpha}} \quad & \sum_{m=1}^M \alpha_m - \frac{1}{2} \sum_{m,n=1}^M \alpha_m \alpha_n y^m y^n (\mathbf{x}^m)^T \mathbf{x}^n \\ \text{s.t.} \quad & \forall m \in \{1, 2, \dots, M\} : 0 \leq \alpha_m \leq \frac{1}{\lambda} \end{aligned} \quad (3.19)$$

This dual form shares some properties with the dual form of GURU. For example, notice that in both cases one tries to maximize the sum of the dual variables α_m . Another issue is that of the norm minimization. The SVM dual explicitly minimizes the norm of the classifier. In our dual, however, the situation is rather implicit: there exist a bound on the norm of the classifier. Without going into the details, we mention that moving a constraint into the objective or vice versa is possible in the context of Lagrangian duality. At last, notice that in spite of the fact that σ and λ play similar roles, increasing λ results in shrinking the feasible region of the SVM dual, whereas in our problem, increasing σ expands the feasible region.

The last issue we discuss in this section is the norm of the optimal classifier. Note that by solving the dual formulation, one can only get the optimal classifier up to a scaling factor. Of course, it is essential to know the norm exactly in order to be able to use the classifier. This goal can be achieved using the following theorem:

Theorem 1.2: The norm of the optimal classifier is

$$\|\mathbf{w}^*\| = \frac{1}{\text{erf}_{inv}(\alpha_m^*) + y^m (\hat{\mathbf{w}}^*)^T \mathbf{x}^m} \quad (3.20)$$

for every m , where $\hat{\mathbf{w}}^*$ is the normalized optimal classifier.

Proof: Equation 3.4 may be written as

$$\begin{aligned} \min_{\mathbf{w}, r, \mathbf{z}} \mathcal{L}(\mathbf{w}, r, \mathbf{z}, \boldsymbol{\alpha}, \lambda, \boldsymbol{\delta}) &= \min_{\mathbf{w}, r} \sum_m \min_{z_m, r_m} \left[r_m f\left(\frac{z_m}{r_m}\right) - \alpha_m z_m - \delta_m r_m \right] \\ &\quad + \lambda [\sigma \|\mathbf{w}\| - r] + \sum_m \alpha_m [1 - y^m \mathbf{w}^T \mathbf{x}^m] - \mu r + r \sum_m \delta_m \\ &= \min_{\mathbf{w}, r} \sum_m \min_{r_m} \left[r_m \min_{z_m} \left[f\left(\frac{z_m}{r_m}\right) - \alpha_m \frac{z_m}{r_m} \right] - \delta_m r_m \right] \\ &\quad + \lambda [\sigma \|\mathbf{w}\| - r] + \sum_m \alpha_m [1 - y^m \mathbf{w}^T \mathbf{x}^m] - \mu r + r \sum_m \delta_m \end{aligned}$$

since $r_m \geq 0$. We define $q_m = \frac{z_m}{r_m}$. Since the equation above depends on z_m only via q_m , we get

$$\begin{aligned} \min_{\mathbf{w}, r, \mathbf{z}} \mathcal{L}(\mathbf{w}, r, \mathbf{z}, \boldsymbol{\alpha}, \lambda, \boldsymbol{\delta}) &= \min_{\mathbf{w}, r} \sum_m \min_{r_m} \left[r_m \min_{q_m} [f(q_m) - \alpha_m q_m] - \delta_m r_m \right] \\ &\quad + \lambda [\sigma \|\mathbf{w}\| - r] + \sum_m \alpha_m [1 - y^m \mathbf{w}^T \mathbf{x}^m] - \mu r + r \sum_m \delta_m \end{aligned}$$

If when substituting the dual optimum in the Lagrangian, there exists a unique primal feasible solution, then it must be primal optimal (see Boyd and Vandenberghe [2004], 5.5.5 for details). Thus, at the optimum $q_m^* = \min_{q_m} [f(q_m) - \alpha_m q_m]$. According to the proof of Lemma 6 it holds that $q_m^* = \text{erf}_{inv}(\alpha_m)$. By exploiting the monotony properties of the problem (that were presented in the beginning of the section), we conclude that

$$\frac{1 - y^m \mathbf{w}^T \mathbf{x}^m}{\sigma \|\mathbf{w}\|} = \text{erf}_{inv}(\alpha_m) \quad (3.21)$$

The desired result follows from basic algebraic operations. ■

Note that the values of the optimal α 's are known, as well as the normalized vector $\hat{\mathbf{w}}^* = \frac{\mathbf{w}^*}{\|\mathbf{w}^*\|}$. Thus, we can compute the optimal norm.

It is possible that the norm of the optimal classifier is bounded (as a function of σ). Although we couldn't prove this result, we conjecture that such a result might stem from a strong duality argument:

$$\|\boldsymbol{\alpha}^*\|_1 = \sum_m \ell_{hinge}^{rob}(\mathbf{x}^m, y^m; \mathbf{w}^*, \sigma^2) \quad (3.22)$$

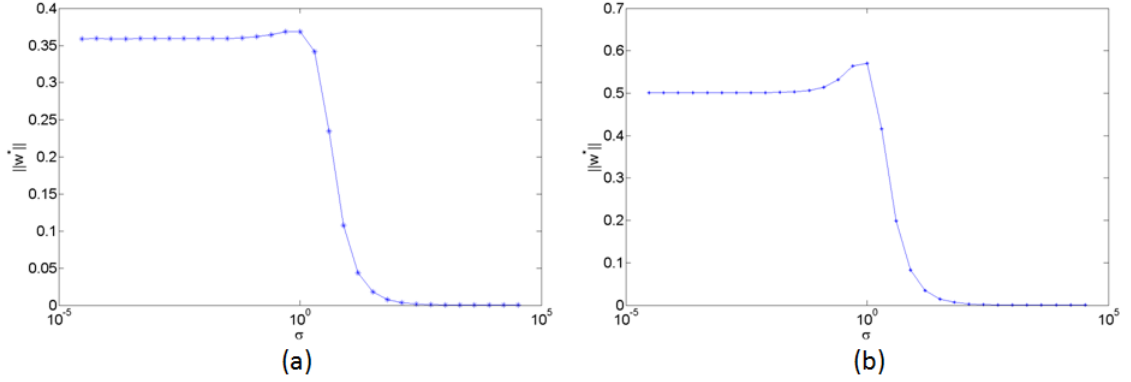


Figure 3.1: Norm of the optimal classifiers trained for the toy problems of Section 5, for various σ values. (a)-(b) represent the respective toy problems.

By plugging Equation 3.21 into the previous equality, we obtain

$$\|\mathbf{w}\| = \frac{\|\boldsymbol{\alpha}^*\|_1}{\sum_m f(\text{erf}_{inv}(\alpha_m))} \quad (3.23)$$

A better understanding of the constraints on $\boldsymbol{\alpha}$ may help bounding the RHS of the equation. We have plotted the norm of the optimal classifiers for the toy problems of Chapter 2 (refer to Section 5 for more details). The results are shown in Figure 3.1 and clearly support this conjecture.

2. A general framework

The dual form we have derived sheds some light on the structure of the problem. In this section we discuss the relation between the loss function f and the norm constraint that appears in the dual. We claim that there is a correspondence between approximations of f and relaxations of the dual problem. More specifically, approximations of the loss function culminates in approximations of the feasible region of the dual problem.

The norm constraint in the dual is a core component of the optimization. We denote by

$$s(\alpha) = \exp\left(-\frac{\text{erf}_{inv}(\alpha)^2}{2}\right) \quad (3.24)$$

the function under summation. It is complicated to handle and understand $s(\alpha)$, thus it is appealing to approximate it using elementary functions. Two such approximations are

$$\begin{aligned} \tilde{s}_1(\alpha) &= H_2(\alpha) = -\alpha \log_2(\alpha) - (1 - \alpha) \log_2(1 - \alpha) \\ \tilde{s}_2(\alpha) &= 4\alpha(1 - \alpha) \end{aligned} \quad (3.25)$$

(see Figure 3.2).

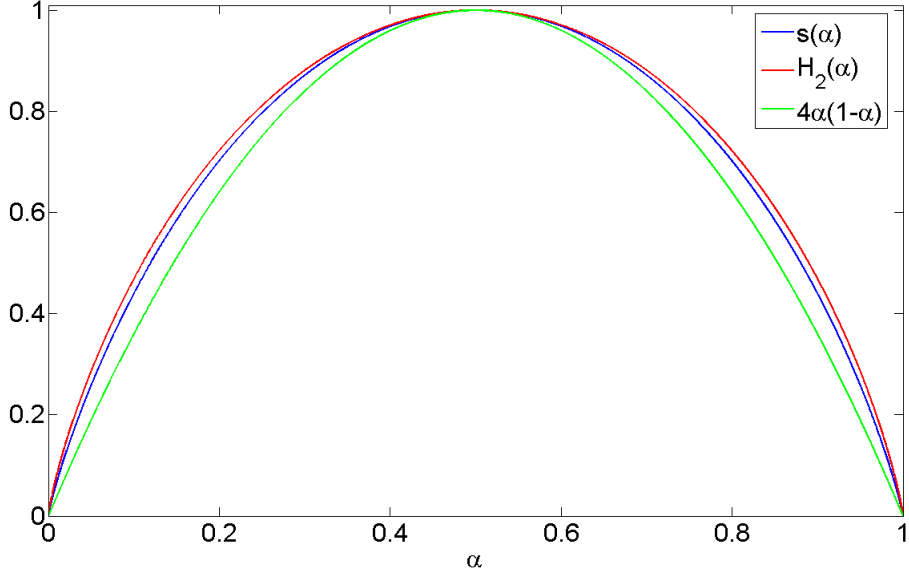


Figure 3.2: The dual constraint may be approximated using elementary functions.

Note that in the previous section we only used f as a means to express g^* (Equation 3.9). Thus, if one replaces f with some alternative convex loss function \tilde{f} , the derivation of the dual will remain correct. Of course, the dual norm constraint will be affected by this change.

In order to understand the nature of the approximations in Equation 3.25, it is necessary to explore the respective dual conjugates.

Lemma 8 Let $\tilde{f}_2(z) = \log_2(1 + 2^z)$. Then its conjugate dual is

$$\tilde{f}_2^*(\alpha) = -\alpha \log_2(\alpha) - (1 - \alpha) \log_2(1 - \alpha) \quad (3.26)$$

Proof: We compute \tilde{f}_2 's conjugate dual:

$$\tilde{f}_2^*(\alpha) = \min_z \tilde{f}_2(z) - \alpha z \quad (3.27)$$

The minimum satisfies:

$$\begin{aligned} \tilde{f}_2'(z) &= \alpha \\ \frac{2^z}{2^z + 1} &= \alpha \\ 2^z &= \frac{\alpha}{1 - \alpha} \\ z &= \log_2 \left(\frac{\alpha}{1 - \alpha} \right) \end{aligned}$$

We plug this equality into the objective and conclude

$$\begin{aligned}\tilde{f}_2^*(\alpha) &= \log_2 \left(1 + \frac{\alpha}{1-\alpha} \right) - \alpha \log_2 \frac{\alpha}{1-\alpha} \\ &= -\alpha \log_2(\alpha) - (1-\alpha) \log_2(1-\alpha)\end{aligned}$$

as claimed. As in the case of our Gaussian robust loss, we have $\alpha \leq 1$. ■

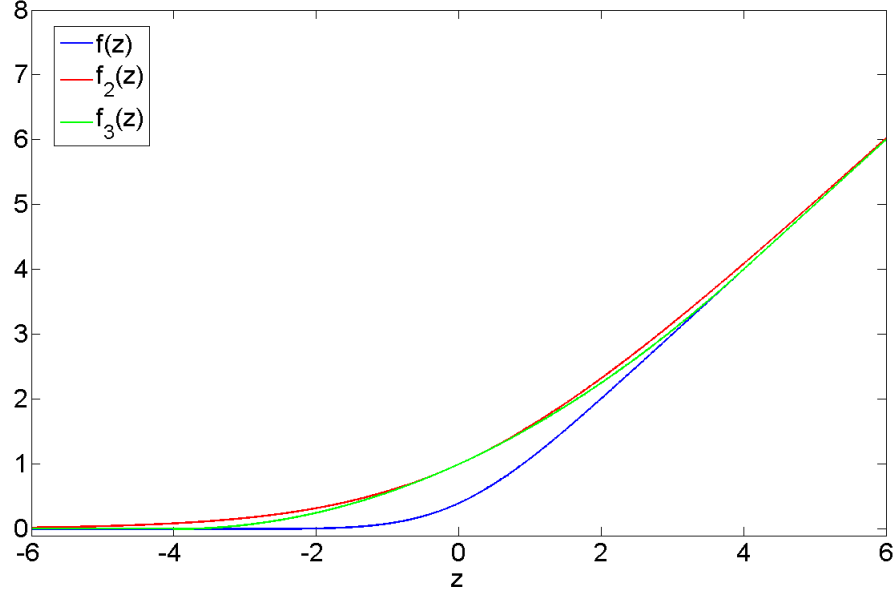


Figure 3.3: Log loss appears naturally in our framework. In addition, we have demonstrated a means to generate some other loss functions, such as the quadratic loss above.

Lemma 9 Let

$$\tilde{f}_3(z) = \begin{cases} 0 & \text{if } z < -4 \\ \frac{(z+4)^2}{16} & \text{if } -4 \leq z \leq 4 \\ z & \text{if } z > 4 \end{cases} \quad (3.28)$$

Then its conjugate dual is

$$\tilde{f}_3^*(\alpha) = \begin{cases} 4\alpha(1-\alpha) & \text{if } 0 \leq \alpha \leq 1 \\ -\infty & \text{if } \alpha > 1 \end{cases} \quad (3.29)$$

Proof: It is easy to verify that \tilde{f}_3 is smooth. We thus compute \tilde{f}_3 's conjugate dual in the following way:

$$\tilde{f}_3^*(\alpha) = \min_z \tilde{f}_3(z) - \alpha z \quad (3.30)$$

Extremum points satisfy:

$$\begin{aligned} \tilde{f}'_3(z) - \alpha &= 0 \\ \begin{cases} -\alpha & \text{if } z < -4 \\ \frac{(z+4)}{8} - \alpha & \text{if } -4 \leq z \leq 4 \\ 1 - \alpha & \text{if } z > 4 \end{cases} &= 0 \end{aligned}$$

The above equation vanishes at $z = 8\alpha - 4$. For $0 \leq \alpha \leq 1$ we have $-4 \leq z \leq 4$, thus we conclude

$$\tilde{f}_3^*(\alpha) \Big|_{0 \leq \alpha \leq 1} = 4\alpha(1 - \alpha)$$

For $\alpha > 1$ we take $z \rightarrow \infty$, and $\tilde{f}_3(z) = \Big|_{z>4} (1 - \alpha)z \rightarrow -\infty$. Altogether we have established the desired result. ■

These lemmas shade some light on ℓ_{hinge}^{rob} and on the structure of our problem. It turns out that the well-known log-loss as well as a quadratic loss that has the same flavour as the Huber loss appear naturally in our framework (see Figure 3.3 for a visualization). What we have demonstrated is that there exist a close connection between approximations of the primal loss and relaxations of the dual problem. Specifically, we have that the dual of

$$\min_{\mathbf{w}} \sum_{m=1}^M \|\mathbf{w}\| \tilde{f} \left(\frac{1 - y^m \mathbf{w}^T \mathbf{x}^m}{\|\mathbf{w}\|} \right) \quad (3.31)$$

is

$$\begin{aligned} \max \quad & \sum_m \alpha_m \\ \text{s.t.} \quad & \left\| \sum_m \alpha_m y^m \mathbf{x}^m \right\| \leq \sum_m \tilde{s}(\alpha_m) \\ & \alpha \geq 0 \end{aligned} \quad (3.32)$$

Note, however, that this connection should be further investigated. It should be observed that not every smooth convex primal loss \tilde{f} yields a perspective that is convex in \mathbf{w} . For that to happen, \tilde{f} should satisfy some mathematical properties that are yet to be understood. One example for such a condition is $\tilde{f}(z) \geq z \frac{d\tilde{f}}{dz}(z)$. Under this condition we can use the same reasoning as in the proof of Theorem 3.2 and conclude that the primal problem is convex. In this case, we can automatically apply the derivation presented in the previous section and deduce the respective dual problem. Another issue that should be better understood is the connection between approximations of f and the robust setup we have begun with. In particular, it is interesting to understand if the logarithmic loss may be interpreted as resulting from RO.

Chapter 4

Introducing Kernels

One of the greatest strengths of the theory of support vector machines, is the simple generalization to nonlinear cases. This generalization is carried out via the elegant notion of kernels. An examination of our derivation suggests that one may apply the kernel trick and introduce a means to learn nonlinear classifiers in Gaussian Robust framework.

In this chapter we will develop a kernelized version of the GURU algorithm. Most of the derivation is straight forward: we begin by giving a representer result. Plugging the new parametrization of the classifier into the framework, we show that our update formulas are perfectly suitable for maintaining this kind of representation. The tricky part stems from the fact that our updates depend directly on the norm of the weights vector. Naive computation of the norm costs $O(M^2)$ operations, which significantly slows down the algorithm. We thus derive a procedure to update the norm in $O(1)$, based on previous computations.

1. A representer result

The first step towards kernelization of GURU, is to change our representation of the classifier from a weights vector (w) to a linear combination of the training samples. The theoretical justification of such an operations is known as a representer result.

The fact that an optimal classifier may be represented as a linear combination of the training sample, stems from the mathematical theory of Hilbert spaces. In our case, as well as in SVM, however, the same result can be derived using far more simple and explicit argumentation. In this section we will show three ways to establish the representer result for the case of GURU. In spite of the fact that we could prove the theorem using abstract argumentation, it is necessary to develop the technical proof, as it lays the foundations for the derivation of the kernelized algorithm.

We start by stating a version of the representer theorem:

Theorem 1.1: Let \mathcal{H} be a reproducing kernel Hilbert space with a kernel $\kappa : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$, a symmetric positive semi-definite function on the compact domain. For any function $L :$

$\mathbb{R}^n \rightarrow \mathbb{R}$, and any nondecreasing function $\Omega : \mathbb{R} \rightarrow \mathbb{R}$. If

$$J^* = \min_{f \in \mathcal{H}} J(f) = \min_{f \in \mathcal{H}} \left\{ \Omega(\|f\|_{\mathcal{H}}^2) + L(f(x_1), f(x_2), \dots, f(x_n)) \right\}$$

is well-defined, then there are some $\alpha_1, \alpha_2, \dots, \alpha_n \in \mathbb{R}$, such that

$$f(\cdot) = \sum_{i=1}^n \alpha_i \kappa(x_i, \cdot) \quad (4.1)$$

achieves $J(f) = J^*$. Furthermore, if Ω is increasing, then each minimizer of $J(f)$ can be expressed in the form of Equation 4.1.

For a proof and more details, see for example Schölkopf and Smola [2002].

As mentioned, we will discuss three techniques to establish the required result. First, using the structure of the updates that GURU perform. Second, by the derivation of the dual problem presented in Chapter 3, and third, using the general representer theorem.

Theorem 1.2: There exists a solution of Equation 2.22 that takes the form

$$\mathbf{w} = \sum_{m=1}^M \alpha_m y^m \mathbf{x}^m \quad (4.2)$$

Proof: Via the structure of GURU

Recall that the updates in the GURU algorithm are of the form

$$\mathbf{w} \leftarrow \mathbf{w} - \frac{\eta}{\sqrt{t}} \left(-y^i \mathbf{x}^i \operatorname{erf} \left(\frac{1 - y^i \mathbf{w}^T \mathbf{x}^i}{\sigma \|\mathbf{w}\|} \right) + \frac{\sigma \mathbf{w}}{\sqrt{2\pi} \|\mathbf{w}\|} \exp \left(-\frac{(1 - y^i \mathbf{w}^T \mathbf{x}^i)^2}{2\sigma^2 \|\mathbf{w}\|^2} \right) \right)$$

It is suggestive to observe that the update formula can be split and written as two successive steps. The first of which is

$$\mathbf{w} \leftarrow \mathbf{w} - \frac{\eta}{\sqrt{t}} \frac{\sigma \mathbf{w}}{\sqrt{2\pi} \|\mathbf{w}\|} \exp \left(-\frac{(1 - y^i \mathbf{w}^T \mathbf{x}^i)^2}{2\sigma^2 \|\mathbf{w}\|^2} \right)$$

followed by

$$\mathbf{w} \leftarrow \mathbf{w} + \frac{\eta}{\sqrt{t}} y^i \mathbf{x}^i \operatorname{erf} \left(\frac{1 - y^i \mathbf{w}^T \mathbf{x}^i}{\sigma \|\mathbf{w}\|} \right) \quad (4.3)$$

The first step is nothing else than a rescaling of the weights vector

$$\mathbf{w} = \gamma \mathbf{w}, \quad \gamma = 1 - \frac{\eta}{\sqrt{t}} \frac{\sigma}{\sqrt{2\pi} \|\mathbf{w}\|} \exp \left(-\frac{(1 - y^i \mathbf{w}^T \mathbf{x}^i)^2}{2\sigma^2 \|\mathbf{w}\|^2} \right) \quad (4.4)$$

Recall that GURU initializes the weight vector as $\mathbf{w} = 0$, which clearly can be represented as

$$\mathbf{0} = \sum_{m=1}^M 0y^m \mathbf{x}^m \quad (4.5)$$

We thus assume that the desired representation exists, and proceed by induction. By plugging the representation into the previous equations, we get

$$\sum_{m=1}^M \alpha_m^{new} y^m \mathbf{x}^m = \sum_{m=1}^M \alpha_m y^m \mathbf{x}^m + \gamma \sum_{m=1}^M \alpha_m y^m \mathbf{x}^m$$

i.e. for all m

$$\alpha_m^{new} = (1 + \gamma) \alpha_m \quad (4.6)$$

where α_m^{new} is the result of the respective update. The second step in the update formula (Equation 4.3), may be written as

$$\sum_{m=1}^M \alpha_m^{new} y^m \mathbf{x}^m = \sum_{m=1}^M \alpha_m y^m \mathbf{x}^m + \mu_i y^i \mathbf{x}^i, \quad \mu_i = \frac{\eta}{\sqrt{t}} y^i \mathbf{x}^i \operatorname{erf} \left(\frac{1 - y^i \mathbf{w}^T \mathbf{x}^i}{\sigma \|\mathbf{w}\|} \right)$$

i.e.

$$\alpha_m^{new} = \begin{cases} \alpha_m & \text{if } m \neq i \\ \alpha_i + \mu_i & \text{if } m = i \end{cases} \quad (4.7)$$

Combining both steps, we end up with the following update rule:

$$\alpha_m^{t+1} = \begin{cases} \gamma \alpha_m^t & \text{if } m \neq i \\ \gamma \alpha_i^t + \mu_i & \text{if } m = i \end{cases} \quad (4.8)$$

Since GURU is guaranteed to converge to the optimum, by taking $t \rightarrow \infty$ we establish the desired result. ■

Proof: Via the dual formulation

We have already seen (Equation 3.4) that

$$\sigma \lambda \frac{\mathbf{w}}{\|\mathbf{w}\|} = \sum_m \alpha_m y^m \mathbf{x}^m$$

By defining $\tilde{\alpha}_m = \frac{\|\mathbf{w}\|}{\sigma \lambda} \alpha_m$ and plugging it into the previous equality, we conclude that

$$\mathbf{w} = \sum_m \tilde{\alpha}_m y^m \mathbf{x}^m$$

as required. ■

Proof: Via the general representer theorem

Set $\Omega \equiv 0$, $L((f(x_1), f(x_2), \dots, f(x_n))) = \sum_{i=1}^n f(x_i)$, $f = \ell_{hinge}^{rob}$. and let κ be the linear kernel $\kappa(x_1, x_2) = x_1^T x_2$. The desired result stems immediately from Theorem 1.1. ■

2. KEN-GURU: A primal kernelized version of GURU

In the previous section we have established a representer result for GURU. The next step in the derivation is to work the components of the algorithm, so the only dependence on the data samples and on the classifier would be via dot products. That being the case, we can apply the kernel trick, namely to replace each dot product $(\mathbf{x}^m)^T \mathbf{x}^n$ with the kernel entry $\kappa(\mathbf{x}^m, \mathbf{x}^n)$ (for details see, for example, Aizerman et al. [1964]; Schölkopf and Smola [2002]). We start by expanding the quantities that appear in the update formula in terms of α_m 's. Then, we introduce a method to update the value of the norm variable in a computationally cheap way. We conclude the section by putting the results together, and presenting the KEN-GURU (**KEr**Nelized **GaUssian RobUst**) algorithm.

In order to compute γ and μ_i of Equation 4.4 and Equation 4.7, one must know the values of $\mathbf{w}^T \mathbf{x}^i$ and $\|\mathbf{w}\|$. Let us expand the first quantity

$$\begin{aligned} \mathbf{w}^T \mathbf{x}^i &= \left(\sum_{m=1}^M \alpha_m y^m \mathbf{x}^m \right)^T \mathbf{x}^i \\ &= \sum_{m=1}^M \alpha_m y^m (\mathbf{x}^m)^T \mathbf{x}^i \\ &= \sum_{m=1}^M \alpha_m y^m K_{mi} \end{aligned}$$

The norm might be computed as

$$\begin{aligned} \|\mathbf{w}\|^2 = \mathbf{w}^T \mathbf{w} &= \left(\sum_{m=1}^M \alpha_m y^m \mathbf{x}^m \right)^T \sum_{n=1}^M \alpha_n y^n \mathbf{x}^n \\ &= \sum_{m=1}^M \sum_{n=1}^M \alpha_m \alpha_n y^m y^n (\mathbf{x}^m)^T \mathbf{x}^n \\ &= \sum_{m=1}^M \sum_{n=1}^M \alpha_m \alpha_n y^m y^n K_{mn} \end{aligned}$$

Note that the Gram matrix K may be precomputed and cached (total cost of $O(M^2)$). Thus, $\mathbf{w}^T \mathbf{x}^i$ can be computed in $O(M)$, and $\|\mathbf{w}\|$ in $O(M^2)$. As both of these values should be computed for each update, the cost of the norm computation is extremely expensive. Instead of computing the norm each time from scratch, it is possible to use its previous value. The updated norm may be computed as

$$\begin{aligned}
\|\mathbf{w}\|_{t+1}^2 &= \sum_{m=1}^M \sum_{n=1}^M \alpha_m^{t+1} \alpha_n^{t+1} y^m y^n K_{mn} \\
&= \sum_{m=1}^M \left[\sum_{n \neq i} \alpha_m^{t+1} \alpha_n^{t+1} y^m y^n K_{mn} + \alpha_m^{t+1} \alpha_i^{t+1} y^m y^i K_{mi} \right] \\
&= \sum_{m=1}^M \sum_{n \neq i} \alpha_m^{t+1} \alpha_n^{t+1} y^m y^n K_{mn} + \sum_{m=1}^M \alpha_m^{t+1} \alpha_i^{t+1} y^m y^i K_{mi} \\
&= \sum_{n \neq i} \left[\sum_{m \neq i} \alpha_m^{t+1} \alpha_n^{t+1} y^m y^n K_{mn} + \alpha_i^{t+1} \alpha_n^{t+1} y^i y^n K_{in} \right] \\
&\quad + \sum_{m=1}^M \alpha_m^{t+1} \alpha_i^{t+1} y^m y^i K_{mi} \\
&= \sum_{n \neq i} \sum_{m \neq i} \alpha_m^{t+1} \alpha_n^{t+1} y^m y^n K_{mn} + \sum_{n \neq i} \alpha_i^{t+1} \alpha_n^{t+1} y^i y^n K_{in} \\
&\quad + \sum_{m \neq i} \alpha_m^{t+1} \alpha_i^{t+1} y^m y^i K_{mi} + \alpha_i^{t+1} \alpha_i^{t+1} y^i y^i K_{ii} \\
&= \sum_{n \neq i} \sum_{m \neq i} \alpha_m^{t+1} \alpha_n^{t+1} y^m y^n K_{mn} + 2 \sum_{m \neq i} \alpha_m^{t+1} \alpha_i^{t+1} y^m y^i K_{mi} \\
&\quad + \alpha_i^{t+1} \alpha_i^{t+1} y^i y^i K_{ii}
\end{aligned}$$

By plugging Equation 4.8 we get

$$\begin{aligned}
\|\mathbf{w}\|_{t+1}^2 &= \gamma^2 \sum_{n \neq i} \sum_{m \neq i} \alpha_m^t \alpha_n^t y^m y^n K_{mn} + 2\gamma \sum_{m \neq i} \alpha_m^t (\gamma \alpha_i^t + \mu_i) y^m y^i K_{mi} \\
&\quad + (\gamma \alpha_i^t + \mu_i)^2 K_{ii} \\
&= \gamma^2 \|\mathbf{w}\|_t^2 + 2\gamma \mu_i y^i \sum_{m=1}^M \alpha_m^t y^m K_{mi} + \mu_i^2 K_{ii} \\
&= \gamma^2 \|\mathbf{w}\|_t^2 + 2\gamma \mu_i y^i \mathbf{w}^T \mathbf{x}^i + \mu_i^2 K_{ii}
\end{aligned}$$

where $\mathbf{w}^T \mathbf{x}^i$ is computed regardless of $\|\mathbf{w}\|^2$. Thus, the value of the norm can be maintained in $O(1)$.

It may be easily observed that the data samples \mathbf{x}^m participate in the computations of the update only via the Gram matrix K . Thus, we can apply the kernel trick, and use

$$K_{ij} = \kappa(\mathbf{x}^i, \mathbf{x}^j) \quad (4.9)$$

for any Mercer Kernel κ . Based on the results established in the previous sections, we may translate GURU into a kernelized version, named KEN-GURU.

We introduce an auxiliary variable ζ , that holds the value of the product $\kappa(\mathbf{w}, \mathbf{x}^i)$ and is evaluated by

$$\zeta_{t+1} = \sum_{m=1}^M \alpha_m^t y^m K(\mathbf{x}^m, \mathbf{x}^i) \quad (4.10)$$

According to Equation 4.4, Equation 4.7 and Equation 4.9 we introduce the following update formulas

$$\gamma_{t+1} = 1 - \frac{\eta}{\sqrt{t}} \frac{\sigma}{\sqrt{2\pi}\nu_t} \exp\left(-\frac{(1 - y^i \zeta_{t+1})^2}{2\sigma^2 \nu_t^2}\right) \quad (4.11)$$

$$\mu_{t+1} = \frac{\eta}{\sqrt{t}} \operatorname{erf}\left(\frac{1 - y^i \zeta_{t+1}}{\sigma \nu_t}\right) \quad (4.12)$$

$$\nu_{t+1} = \sqrt{\gamma_{t+1}^2 \nu_t^2 + 2\gamma_{t+1} \mu_{t+1} y^i \zeta_{t+1} + \mu_{t+1}^2 K_{ii}} \quad (4.13)$$

Algorithm 2: KEN-GURU($\kappa, \mathcal{S}, \eta_0, \epsilon$)

Data: Kernel function κ , training set \mathcal{S} , learning rate η_0 , accuracy ϵ

Result: α

//initializations

forall $m, n = 1..m$ **do**

 | $K_{mn} = \kappa(\mathbf{x}^m, \mathbf{x}^n)$

end

$\alpha^0 \leftarrow \mathbf{0}$;

$\nu_0 \leftarrow 0$;

$t \leftarrow 0$;

while $\Delta L \geq \epsilon$ **do**

 //randomize a sample

$i \leftarrow \text{rand}(M)$;

 //evaluate coefficients

 Compute ζ_{t+1} (Equation 4.10);

 Compute γ_{t+1} (Equation 4.11);

 Compute μ_{t+1} (Equation 4.12);

 //update alphas

$\alpha_{t+1} \leftarrow \gamma_{t+1} \alpha_t$;

$\alpha_{t+1}^i \leftarrow \alpha_{t+1}^i + \mu_{t+1}$;

$t \leftarrow t + 1$;

end

return α ;

The correctness of the algorithm stems directly from that of GURU.

Name	GURU(%)	SVM(%)
Ionosp- here	83.55	81.58
diabetes	68.59	66.67
splice	92.28	92.28
1 vs. 2		
USPS	97.86	98
3 vs. 5		
USPS	98.29	98.71
5 vs. 8		
USPS	98.43	97.86
7 vs. 9		

Table 4.1: Results summary for KEN-GURU.

3. Experiments

In this section we present experimental results regarding the performance of KEN-GURU. We show how σ affects the learned classifier and then compare KEN-GURU to SVM on USPS pairs and on the Ionosphere database (see Table 2.1 for details). For the USPS tasks, a polynomial kernel of degree 2 was used and for Ionosphere, RBF with $\gamma = 1$. The results are summarized in Table 4.1.

Consider Figure 4.1, in which KEN-GURU classifiers trained for various values of the parameter σ with a polynomial kernel of degree 2 are presented. The toy problem was synthesized by first generating uniformly points on $[-7.5, 7.5] \times [-7.5, 7.5]$. Points which fall within the ball of radius 2 around the origin were assigned a positive label. Points which are more distant from the origin than 3.5 units were taken as negative examples. Points which fell in between were dropped. Observe that increasing σ puts extra emphasis on the number of samples in each class. Specifically, in the problem at hand, there are much more points outside the circle than inside. When σ is rather small, the training is ‘local’ in the sense that each sample governs what happens in its immediate environment. On the contrary, when σ is relatively big, the emphasis is on global tendencies.

On the Ionosphere database, KEN-GURU performs significantly better than SVM. Recall that the outperformance of GURU on SVM in this case is consistent with the performance in the case of a linear kernel. This behavior is explained by the noisy nature of the Ionosphere database. For the USPS couples, KEN-GURU’s performance is pretty similar to that of SVM.

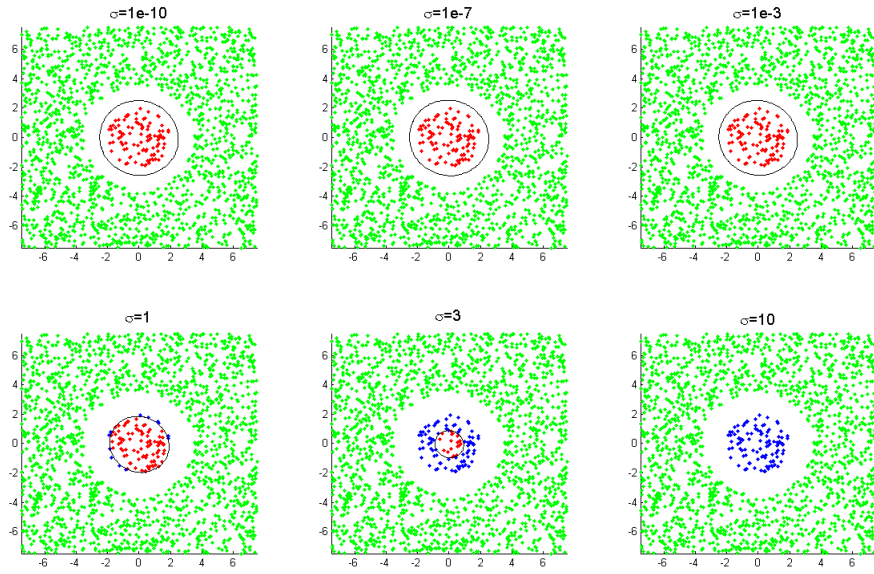


Figure 4.1: KEN-GURU performance on a radial data set. The green and red points indicate data points that were correctly classified (each color stands for one of the classes). Blue points indicate misclassification. The parameter σ determines how distant is the effect of each data point. Note that for small values of σ , the behavior of the classifier is determined locally by the samples. For rather big σ , the effect is global, in the sense that the behavior of the classifier is determined by close as well as distant data samples.

Chapter 5

The Multiclass Case

In the previous chapters we have developed the binary algorithm GURU, and its kernelized version KEN-GURU. In this chapter we will analyze another extension of the algorithm, for the case of multiclass cases.

The ideas that were presented in Chapter 2 may be generalized for the multi-class case. To that end, we first should generalize the loss function we are working with. This goal is achieved by solving the generalized problem of the adversarial choice. After establishing this result we devise the effective robust loss function, and devise an optimization algorithm for it.

We relax the problem twice in order to solve it. First, we work with the sum-of-hinges loss function (Weston and Watkins [1999]). In addition, we use a superset of noise distribution, that contains all covariance matrix with a bounded maximal eigenvalue. By the end of the chapter we will prove that for the binary case the maximal eigenvalue and trace constraint give the same result.

The setting we address in the followings is of data drawn from $\mathcal{X} = \mathbb{R}^d$, accompanied by labels drawn from $\mathcal{Y} = \{1, 2, \dots, C\}$. The learning task is to train the weight vectors $\mathbf{w}_1, \mathbf{w}_2, \dots, \mathbf{w}_C$. The target classifier is $\phi : \mathcal{X} \rightarrow \mathcal{Y}$, defined by

$$\phi(\mathbf{x}; \mathbf{w}_1, \mathbf{w}_2, \dots, \mathbf{w}_C) = \max_{y \in \mathcal{Y}} [\mathbf{w}_y^T \mathbf{x}] \quad (5.1)$$

1. Problem formulation

In this section we formally describe the generalization of the learning task from the binary to the multiclass case. We show that the generalization culminates in a loss function which is the sum of several appropriate binary losses.

In Chapter 2 we have started our derivation from the hinge loss

$$\ell_{\text{hinge}}(\mathbf{x}^m, y^m; \mathbf{w}) = [1 - y^m \mathbf{w}^T \mathbf{x}]_+ \quad (5.2)$$

The most common generalization of the hinge loss to the multi-class case is

$$\ell_{\text{mult}}(\mathbf{x}^m, y^m; \mathbf{w}_1, \dots, \mathbf{w}_C) = \max_y [\mathbf{w}_y^T \mathbf{x}^m - \mathbf{w}_{y^m}^T \mathbf{x}^m + \delta_{y, y^m}] \quad (5.3)$$

However, this loss function is not applicable in our framework (see Appendix C). Instead, we suggest to minimize the following surrogate loss function (Weston & Watkins, e.g. ref):

$$\ell_{sum}(\mathbf{x}^m, y^m; \mathbf{w}_1, \mathbf{w}_2, \dots, \mathbf{w}_C) = \sum_{i \neq y} [1 - (\mathbf{w}_{y^m} - \mathbf{w}_i)^T \mathbf{x}^m]_+ \quad (5.4)$$

which is a surrogate to the zero-one loss.

Let us write down the formulation of the problem in this case:

$$\min_{\mathbf{w}_1, \mathbf{w}_2, \dots, \mathbf{w}_C} \sum_m \max_{\Sigma \in \Gamma_\beta} \mathbb{E}_{\mathbf{n} \sim \mathcal{N}(\mathbf{0}, \Sigma)} \sum_{y' \neq y^m} [1 - (\mathbf{w}_{y^m} - \mathbf{w}_{y'})^T (\mathbf{x}^m + \mathbf{n})]_+ \quad (5.5)$$

where

$$\Gamma_\beta = \{\Sigma \in \mathbf{PSD} \mid \rho(\Sigma) \leq \beta\} \quad (5.6)$$

and ρ is the spectral norm of a matrix, defined by

$$\rho(A) = \sqrt{\lambda_{\max}(A^* A)}$$

Using this set we constrain the maximal power of noise that the adversary may spread in each primary direction.

2. The adversarial choice

In the followings we will focus on deriving the adversarial choice for the problem at hand. It appears that in the current setup, the solution is simpler than the one we had in Chapter 2.

2.1 Applying a spectral norm constraint

Let us investigate what is the adversary's optimal way for spreading the noise. The ideas of the development are similar to that of Theorem 2.1.

Denote

$$\Delta \mathbf{W}_{y, y'} = \mathbf{w}_y - \mathbf{w}_{y'} \quad (5.7)$$

Using the same procedure we have employed in the binary case (see Section 2.1 and Equation 2.15 thereby) we can write Equation 5.5 as:

$$\min_{\mathbf{w}_1, \mathbf{w}_2, \dots, \mathbf{w}_C} \sum_m \max_{\Sigma \in \Gamma_\beta} \sum_{y' \neq y^m} L(\mathbf{x}^m, +1; \Delta \mathbf{W}_{y^m, y'}, \Delta \mathbf{W}_{y^m, y'}^T \Sigma \Delta \mathbf{W}_{y^m, y'}) \quad (5.8)$$

i.e. the task at hand is to optimize the effective loss function

$$\ell_{sum}^{rob}(\mathbf{x}^m, y^m; \mathbf{w}_1, \mathbf{w}_2, \dots, \mathbf{w}_C, \beta) = \max_{\Sigma \in \Gamma_\beta} \sum_{y' \neq y^m} L(\mathbf{x}^m, +1; \Delta \mathbf{W}_{y^m, y'}, \Delta \mathbf{W}_{y^m, y'}^T \Sigma \Delta \mathbf{W}_{y^m, y'}) \quad (5.9)$$

Observe that in every appearance of ℓ_{hinge}^{rob} , the label y^m was replaced with $+1$. The reason for this change is that we are classifying using the weight vector $\mathbf{w}_{y^m} - \mathbf{w}_{y'}$. That is, our prediction is

$$(\mathbf{w}_{y^m} - \mathbf{w}_{y'})^T \mathbf{x}^m = \mathbf{w}_{y^m}^T \mathbf{x}^m - \mathbf{w}_{y'}^T \mathbf{x}^m$$

Our objective is, of course, to have $\mathbf{w}_{y^m}^T \mathbf{x}^m > \mathbf{w}_{y'}^T \mathbf{x}^m$, which corresponds to the label $+1$. The next theorem specifies the adversarial choice of the covariance matrix Σ , and is the multi-class analog of Theorem 2.1:

Theorem 2.1: The optimal Σ in Equation 5.9 is given by $\Sigma^* = \beta I$.

Proof: In Lemma 2 we have shown that L is monotone increasing in its 4^{th} argument. By the Cauchy-Schwartz inequality we have that

$$\Delta \mathbf{W}_{y^m, y'}^T \Sigma \Delta \mathbf{W}_{y^m, y'} \leq \beta \|\Delta \mathbf{W}_{y^m, y'}\|^2 \quad (5.10)$$

On the other hand, it holds that for all y'

$$\Delta \mathbf{W}_{y^m, y'}^T \beta I \Delta \mathbf{W}_{y^m, y'} = \beta \|\Delta \mathbf{W}_{y^m, y'}\|^2 \quad (5.11)$$

hence this upper bound is attained for all $C - 1$ summands concurrently with $\Sigma = \beta I$. The geometric interpretation of this result is that under the spectral norm constraint, the adversary will choose to spread the noise in an isotropic fashion around the sample point. ■

We thus get the following optimization problem:

$$\min_{\mathbf{w}_1, \mathbf{w}_2, \dots, \mathbf{w}_C} \sum_m \sum_{y' \neq y^m} L(\mathbf{x}^m, +1; \Delta \mathbf{W}_{y^m, y'}, \beta \|\Delta \mathbf{W}_{y^m, y'}\|^2) \quad (5.12)$$

Applying the same terminology used in the binary case, we have:

$$\min_{\mathbf{w}_1, \mathbf{w}_2, \dots, \mathbf{w}_C} \sum_m \sum_{y' \neq y^m} \ell_{hinge}^{rob}(\mathbf{x}^m, +1; \Delta \mathbf{W}_{y^m, y'}, \beta) \quad (5.13)$$

and Equation 5.9 equals

$$\ell_{sum}^{rob}(\mathbf{x}^m, y^m; w_1, w_2, \dots, w_C, \beta) = \sum_{y' \neq y^m} \ell_{hinge}^{rob}(\mathbf{x}^m, +1; \Delta \mathbf{W}_{y^m, y'}, \beta) \quad (5.14)$$

2.2 The connection to the trace constraint

It is interesting to examine the reduction of the multiclass loss we have derived, to the binary case. Note that since we have used a substantially larger matrix collection, there is no apriori reason to expect that the results will coincide.

Taking $C = 2$ brings us back to the binary case. We use \mathbf{w}_{+1} , \mathbf{w}_{-1} for the weight vectors of the classes. By expanding Equation 5.9, we get

$$\ell_{sum}^{rob}(\mathbf{x}^m, y^m; \mathbf{w}_{+1}, \mathbf{w}_{-1}, \beta) = \ell_{hinge}^{rob}(\mathbf{x}^m, +1; \mathbf{w}_{y^m} - \mathbf{w}_{-y^m}, \beta) \quad (5.15)$$

If we take $\mathbf{w} = \mathbf{w}_{+1} - \mathbf{w}_{-1}$, we end up with

$$\ell_{sum}^{rob}(\mathbf{x}^m, y^m; \mathbf{w}_{+1}, \mathbf{w}_{-1}, \beta) = \ell_{hinge}^{rob}(\mathbf{x}^m, y^m; \mathbf{w}, \beta) \quad (5.16)$$

It is interesting to observe that the resulting loss functions are identical, even though the constraints we put on the covariance matrices are different. In order to explain this phenomenon, let us go back the geometric intuition that we have given prior to the proof of Theorem 2.1.

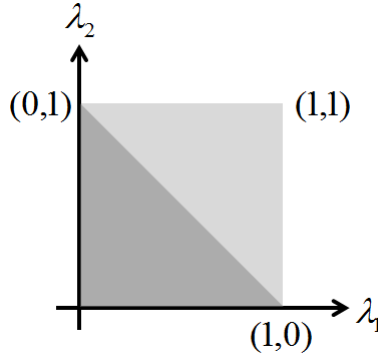


Figure 5.1: Visualization of Λ_1 and Γ_1 in the 2-dimensional case. The axes represent the eigenvalues of Σ . The dark shaded region contains all the matrices having $\lambda_1 + \lambda_2 \leq 1$, i.e. corresponds to Λ_1 . The light area corresponds to Γ_1 , and consists of all the matrices with $\max\{\lambda_1, \lambda_2\} \leq 1$.

Consider Figure B.1, which presents a visualization of Λ_β and Γ_β in the 2-dimensional case. What we have shown in Theorem 2.1, is that the multiclass adversary will choose the point $(1, 1)$. Under the trace constraint, however, the adversary will have to choose either $(1, 0)$, $(0, 1)$, or any other point lying on the line connecting them. Our geometric intuition says that all the power that was not spread perpendicularly to the separating hyperplane is irrelevant. Thus, when the adversary has to choose a directional noise, he would take the perpendicular direction. On the other hand, if we limit his action axis-wise (and not overall), he will surely choose to spread the noise equally over all of the axes.

3. M-GURU: a primal algorithm for the multiclass case

In the following we generalize GURU (that was presented in Section 4) for the multiclass case. As a direct corollary of the results presented in previous chapters, we have that our loss function in this case is strictly-convex. Thus, we turn to devise an SGD procedure.

We shall begin by computing the gradient of $\ell_{sum}^{rob}(\mathbf{x}^m, y^m; \mathbf{w}_1, \mathbf{w}_2, \dots, \mathbf{w}_C, \beta)$. For convenience, we write it in terms of the binary loss function ℓ_{hinge}^{rob} :

$$\nabla_{\mathbf{w}_r} \ell_{sum}^{rob}(\mathbf{x}^m, y^m; \mathbf{w}_1, \mathbf{w}_2, \dots, \mathbf{w}_C, \beta) = \begin{cases} \sum_{y' \neq r} \nabla_{\mathbf{w}} \ell_{hinge}^{rob}(\mathbf{x}^n, +1; \mathbf{w}, \beta) \Big|_{\mathbf{w}=\mathbf{w}_{y^m}-\mathbf{w}_{y'}} & \text{if } r = y^m \\ -\nabla_{\mathbf{w}} \ell_{hinge}^{rob}(\mathbf{x}^n, +1; \mathbf{w}, \beta) \Big|_{\mathbf{w}=\mathbf{w}_{y^m}-\mathbf{w}_r} & \text{otherwise} \end{cases}$$

Following the considerations that we have introduced in Section 4, we devise an SGD procedure for the minimization task:

Algorithm 3: M-GURU($\mathcal{S}, \eta_0, \epsilon$)

Data: Training set \mathcal{S} , learning rate η_0 , accuracy ϵ

Result: w

$w \leftarrow \mathbf{0}$;

while $\Delta L \geq \epsilon$ **do**

$m \leftarrow \text{rand}(M)$;

for $y' \in \{1, 2, \dots, C\}$ **do**

$\mathbf{w}_{y'} \leftarrow \mathbf{w}_{y'} - \frac{\eta_0}{\sqrt{t}} \nabla_{\mathbf{w}_{y'}} \ell_{sum}^{rob}(\mathbf{x}^m, y^m; \mathbf{w}_1, \mathbf{w}_2, \dots, \mathbf{w}_C, \beta)$;

end

end

return w ;

In Algorithm 3, the notion of stochastic gradient was applied once, to the extent that our updates depend on a single sample in each iteration. It may be applied again, however. Instead of updating all the weight vectors concurrently, one might randomize which vector to update, as well. The resulting algorithm is

Algorithm 4: M-GURU- S^2 ($\mathcal{S}, \eta_0, \epsilon$)

Data: Training set \mathcal{S} , learning rate η_0 , accuracy ϵ

Result: w

$w \leftarrow \mathbf{0}$;

while $\Delta L \geq \epsilon$ **do**

$m \leftarrow \text{rand}(M)$;

$y' \leftarrow \text{rand}(C)$ $\mathbf{w}_{y'} \leftarrow \mathbf{w}_{y'} - \frac{\eta_0}{\sqrt{t}} \nabla_{\mathbf{w}_{y'}} \ell_{sum}^{rob}(\mathbf{x}^m, y^m; \mathbf{w}_1, \mathbf{w}_2, \dots, \mathbf{w}_C, \beta)$;

end

return w ;

Name	#Training samples	#Cross- validation samples	#Test samples	#features	#classes
Toy-3	200	200	200	2	3
Toy-4	200	200	200	2	4
USPS 3,5,8	1200	1050	1050	256	3
USPS 0-9	3000	2000	6000	256	10
splice	1000	1000	1190	60	3
wine	50	50	78	13	3

Table 5.1: Description of the databases used in the binary case

4. Experiments

M-GURU and M-GURU- S^2 were tested on toy problems, USPS and a couple of UCI databases (Frank and Asuncion [2010]). The datasets are detailed in Table 5.1. In Toy-3 and Toy-4 each class is a Gaussian distribution. These problems are visualized in Figure 5.3. The results are summarized in Table 5.2.

Observe that the performance of M-GURU is similar to that of SVM. Nonetheless, it should be noted that SVM slightly outperforms M-GURU. This difference is explained by the fact that M-GURU is based on the sum-of-hinges loss function, which is a looser surrogate of the zero-one loss than the SVM multi-hinge loss function. We have tested the relative performance of M-GURU and M-GURU- S^2 on the toy-3 dataset.

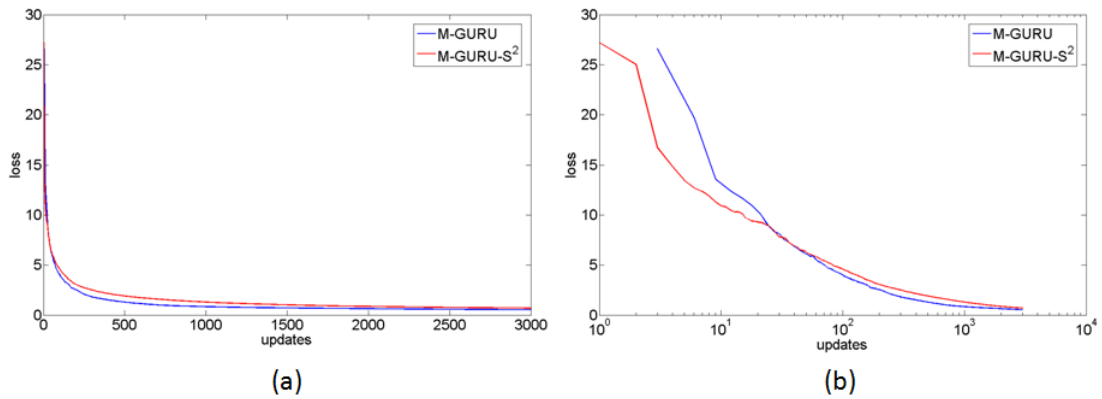


Figure 5.2: A typical run of M-GURU and M-GURU- S^2 on the toy-3 dataset. The loss is plotted against the number of updates that were performed. The S^2 variant appears to have an advantage in the descent phase. In the convergence phase, however, M-GURU takes the lead. Overall, the performance of both variants is pretty similar. (a) linear scale. (b) semi-logarithmic scale.

Name	M-GURU(%)	M-GURU- S^2 (%)	SVM(%)
Toy-3	98.67	98	98.67
Toy-4	96	96	96
USPS	94.67	94.57	94.857
3,5,8			
USPS	92.78	92.7	92.85
0-9			
splice	89.08	89.08	89.5
wine	92.31	91.03	92.31

Table 5.2: Summary of the results.

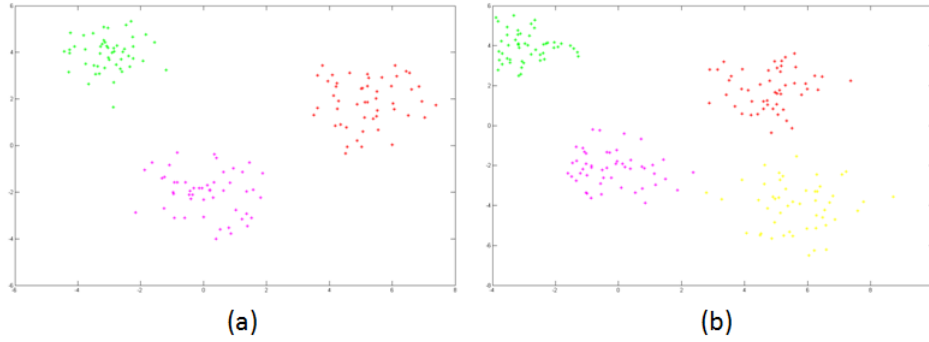


Figure 5.3: The toy problems used in the testing of M-GURU and M-GURU- S^2 . (a) Toy-3. (b) Toy-4.

We observe that M-GURU outperforms the S^2 variant. Our experiments show that the empirical behavior of the classifiers stabilizes a significant time before the optimization process converges. Thus, M-GURU- S^2 may be used to learn classifiers more quickly.

Chapter 6

Discussion

1. Contribution

In this work we presented a new robust learning framework. In our framework we minimize the expected loss over a spreading of the sample points. Each displacement is assumed to take place with a probability that depends on its distance from the original point. Thus, we effectively replace each point with a fading cloud.

We have analyzed the case of Gaussian noise distribution, where the underlying loss measure is the hinge-loss. In this case, we have shown that the resulting effective loss function is a smooth strictly-convex upper-approximation of the hinge-loss, denoted ℓ_{hinge}^{rob} . One of the main advantages of this loss function, is its parameter σ that has a clear meaning: the variance of the noise that contaminates the data. Similarly to SVM, our algorithm, named GURU, depends on a single parameter. A significant difference is the ability to assign a value to this parameter. In the case of SVM, for a long time all that was known on this parameter is that it controls the tradeoff between the training error and the margin of the classifier. Xu et al. [2009] have shown that SVM is equivalent to a robust formulation in which the parameter corresponds to the radius of a rigid ball in which the sample point may be displaced. This result, however, relates the parameter with the entire data set. Thus, it is still difficult to tune it. In our method, σ is the magnitude of noise that possibly corrupts each sample point, hence it might be evaluated from physical consideration, such as the process that generates the data, etc. Without putting extra effort, we are able to point out an alternative explanation for non-regularized SVMs lack of ability to generalize. We have shown that as σ tends to 0, ℓ_{hinge}^{rob} coincides asymptotically with the hinge loss. Thus, non-regularized SVM may be understood as not trying to achieve robustness to perturbations, hence it tends to overfit the data. We have shown that ℓ_{hinge}^{rob} may be written as a perspective of a smooth loss function (denoted f), where the scaling factor is $\sigma\|w\|$. This representation suggests that the robust framework we have developed introduces a multiplicative regularization. Using both this representation we have derived a dual problem. The dual formulation depends on the actual loss function f only via its conjugate dual. Thus, it is possible to plug into the same formulation some other losses that follow certain conditions. In particular, as we have demonstrated in Chapter 3, there is

a tight connection between approximations of the loss function and relaxations of the dual problem. We believe that applying the same technique we have applied here to other loss functions will result in new robust learning algorithms. The connection between the primal loss and the resulting dual should be investigated more thoroughly. The algorithmic approach we have taken in this work is rather simplistic. Due to the fact that our objective is strictly-convex, many off-the-shelf convex optimization algorithms may be used. Our method of choice was stochastic gradient descent. Furthermore, if there is a bound on the norm of the optimal classifier (as in SVM. see Shalev-Shwartz et al. [2007a] for details), it is probably possible to use it in order to achieve even faster algorithms. Specifically, subject to such a bound, we may restrict the optimization problem to a ball around the origin. In this ball, it is possible that our loss function is strongly-convex, hence it can be optimized using more aggressive procedure (Shalev-Shwartz and Kakade [2008]). Our generalization to Mercer kernels, is done based on the primal formulation. In order to compute the updates fast ($O(M)$), we have shown how to maintain the value of the norm of the classifier in $O(1)$ based on pre-computed values. This technique may be employed in Pegasos, e.g, in order to perform the projection step efficiently.

2. Generalizations

The framework we have introduced may be generalized in couple of interesting directions. Obviously, various families of noise distributions may be plugged into the model. One particularly interesting is the class of all probability distributions having a specific first and second moment. Vandenberghe et al. [2007] have shown that the probability of a set defined by quadratic inequalities may be computed using semidefinite programming. In addition, they have shown that the optimum is achieved over a discrete probability distribution. We conjecture that a similar technique may be employed in our case, in order to show that the optimum of the loss expectation is attained over a discrete distribution. In addition, the same framework can be used in order to explore more convex perturbations. For example, in the field of computer vision it is possible to assume that the adversary rotates or translates the sample, and that the distribution of these perturbations is chosen adversely. In order to make this practical, it is crucial to understand in which cases the integration and integration of the loss are possible.

Regarding the theoretical aspects of this work, it still remains to show how to derive performance bounds for the introduced framework. In particular, it is interesting to understand what kind of guarantees can be derived for the general perspective-optimization framework we have discussed.

Bibliography

- M. A. Aizerman, E. A. Braverman, and L. Rozonoer. Theoretical foundations of the potential function method in pattern recognition learning. In *Automation and Remote Control*, number 25, pages 821–837, 1964.
- Shun-Ichi Amari. Natural gradient works efficiently in learning. *Neural Comput.*, 10: 251–276, February 1998. ISSN 0899-7667. doi: 10.1162/089976698300017746. URL <http://portal.acm.org/citation.cfm?id=287476.287477>.
- Jean baptiste Pothin and Cdric Richard. Incorporating prior information into support vector machines in the form of ellipsoidal knowledge sets. 2008.
- Peter L. Bartlett, Olivier Bousquet, and Shahar Mendelson. Local rademacher complexities. In *Annals of Statistics*, pages 44–58, 2002.
- Dimitri P. Bertsekas, Angelia Nedic, and Asuman E. Ozdaglar. *Convex analysis and optimization*. Athena Scientific, Nashua, USA, 2003.
- Chiranjib Bhattacharyya, L. R. Grate, Michael I. Jordan, Laurent El Ghaoui, and I. Saira Mian. Robust sparse hyperplane classifiers: Application to uncertain molecular profiling data. *Journal of Computational Biology*, 11(6):1073–1089, 2004a.
- Chiranjib Bhattacharyya, Pannagadatta K. Shivaswamy, and Alex J. Smola. A second order cone programming formulation for classifying missing data. In *NIPS*, 2004b.
- Jinbo Bi and Tong Zhang. Support vector classification with input data uncertainty. *nips*, 2004.
- Chris M. Bishop. Training with noise is equivalent to tikhonov regularization. *Neural Computation*, 7:108–116, 1994.
- Léon Bottou and Olivier Bousquet. The Tradeoffs of Large Scale Learning. In J. C. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems 20*, pages 161–168, 2008. URL http://books.nips.cc/papers/files/nips20/NIPS2007_0726.bib.

- Léon Bottou and Yann LeCun. Large scale online learning. In Sebastian Thrun, Lawrence K. Saul, and Bernhard Schölkopf, editors, *NIPS*. MIT Press, 2003. ISBN 0-262-20152-6.
- Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004. ISBN 0521833787. URL <http://www.stanford.edu/~boyd/cvxbook/>.
- Olivier Chapelle. Training a support vector machine in the primal. *Neural Computation*, 19:1155–1178, 2007.
- Koby Crammer and Yoram Singer. On the algorithmic implementation of multiclass kernel-based vector machines. *The Journal of Machine Learning Research*, 2:265–292, 2002. ISSN 1532-4435.
- Theodoros Evgeniou, Massimiliano Pontil, and Tomaso Poggio. Regularization networks and support vector machines. In *Advances in Computational Mathematics*, pages 1–50. MIT Press, 2000.
- A. Frank and A. Asuncion. UCI machine learning repository, 2010. URL <http://archive.ics.uci.edu/ml>.
- Laurent El Ghaoui and Herve Lebre. Robust solutions to least-squares problems with uncertain data, 1997.
- Amir Globerson and Sam T. Roweis. Nightmare at test time: robust learning by feature deletion. In William W. Cohen and Andrew Moore, editors, *ICML*, volume 148 of *ACM International Conference Proceeding Series*, pages 353–360. ACM, 2006. ISBN 1-59593-383-2.
- Jyrki Kivinen, Alexander J. Smola, and Robert C. Williamson. Online learning with kernels, 2003.
- Daphne Koller, Dale Schuurmans, Yoshua Bengio, and Léon Bottou, editors. *Advances in Neural Information Processing Systems 21, Proceedings of the Twenty-Second Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 8-11, 2008*, 2009. MIT Press.
- Angelia Nedic and Dimitri Bertsekas. Convergence rate of incremental subgradient algorithms. In *Stochastic Optimization: Algorithms and Applications*, pages 263–304. Kluwer, 2000.
- Alexei Pozdnoukhov, Samy Bengio, Alexei Pozdnoukhov, and Samy Bengio. A kernel classifier for distributions, 2005.
- Andrew M. Ross. Useful bounds on the expected maximum of correlated normal variables, 2003.

- Bernhard Schölkopf and Alexander J. Smola. *Learning with kernels : support vector machines, regularization, optimization, and beyond*. Adaptive computation and machine learning. MIT Press, 2002. URL <http://www.worldcat.org/oclc/48970254>.
- Shai Shalev-Shwartz and Sham M. Kakade. Mind the duality gap: Logarithmic regret algorithms for online optimization. In Koller et al. [2009], pages 1457–1464.
- Shai Shalev-Shwartz, Yoram Singer, and Nathan Srebro. Pegasos: Primal estimated sub-gradient solver for svm. In Zoubin Ghahramani, editor, *ICML*, volume 227 of *ACM International Conference Proceeding Series*, pages 807–814. ACM, 2007a. ISBN 978-1-59593-793-3.
- Shai Shalev-Shwartz, Yoram Singer, and Nathan Srebro. Pegasos: Primal estimated sub-gradient solver for svm. 2007b. URL <http://ttic.uchicago.edu/~shai/papers/ShalevSiSr07.pdf>. A fast online algorithm for solving the linear svm in primal using sub-gradients.
- Pannagadatta K. Shivaswamy, Chiranjib Bhattacharyya, and Alexander J. Smola. Second order cone programming approaches for handling missing and uncertain data. *Journal of Machine Learning Research*, 7:1283–1314, 2006.
- Alexander Smola, Bernhard Schölkopf, Rudower Chaussee, and Bernhard Schölkopf. From regularization operators to support vector kernels. In *In Advances in Neural information processings systems 10*, pages 343–349. MIT Press, 1998.
- Lieven Vandenberghe, Stephen Boyd, and Katherine Comanor. Generalized chebyshev bounds via semidefinite programming. *SIAM Review*, 49, 2007.
- Vladimir Vapnik. *The Nature of Statistical Learning Theory*. Springer, New York, 1995.
- J. Weston and C. Watkins. Support vector machines for multi-class pattern recognition. In *Proceedings of the Seventh European Symposium On Artificial Neural Networks*, volume 4. Citeseer, 1999.
- Huan Xu, Constantine Caramanis, and Shie Mannor. Robust regression and lasso. In Koller et al. [2009], pages 1801–1808.
- Huan Xu, Constantine Caramanis, and Shie Mannor. Robustness and regularization of support vector machines. *Journal of Machine Learning Research*, 10:1485–1510, 2009.
- Jian Zhang, Rong Jin, Yiming Yang, and Alexander G. Hauptmann. Modified logistic regression: An approximation to svm and its applications in large-scale text categorization. In Tom Fawcett and Nina Mishra, editors, *ICML*, pages 888–895. AAAI Press, 2003. ISBN 1-57735-189-4.

Appendix A

Single-Point Algorithms

The object of this work is to learn classifiers that are robust to noise. As discussed, a possible way to achieve this goal is by applying an adversarial framework. The most important issue in this case is designing an effective adversary. While in the previous chapters of the work we explored more sophisticated adversaries, it is nice to end the journey with a rather simple mathematical formulation. The binary version of the algorithms was extensively studied. We review the result here for the sake of a complete presentation. A simple generalization for the multiclass case is presented subsequently.

1. Problem presentation

Maybe the simplest action that the adversary can take at test-time is displacing a test point, in such a way that will cause this point to be missclassified. If we limit the freedom given to the adversary, it might not be able to corrupt the classification of the point, but rather only reduce the associated confidence. The model that we will explore in the followings grants the adversary the ability to displace a sample point within a ball centered at the original point.

In order for the learned classifier to be robust to such displacements, we should modify the objective of the learning task. In the following we present and analyze one way to do it, by optimizing the worst-case scenario:

$$\min_{\mathbf{w}} \max_{\|\Delta \mathbf{x}^m\| \leq \delta: m=1..M} \frac{\lambda}{2} \|\mathbf{w}\|^2 + \sum_{m=1}^M [1 - y^m \mathbf{w}^T (\mathbf{x}^m + \Delta \mathbf{x}^m)]_+ \quad (\text{A.1})$$

This formulation has an additive structure, in which each term $\Delta \mathbf{x}^m$ appears exactly once. We use these properties in order to decouple the optimization problem. The learning task at hand in this case is thus

$$\min_{\mathbf{w}} \frac{\lambda}{2} \|\mathbf{w}\|^2 + \sum_{m=1}^M \max_{\|\Delta \mathbf{x}^m\| \leq \delta} [1 - y^m \mathbf{w}^T (\mathbf{x}^m + \Delta \mathbf{x}^m)]_+ \quad (\text{A.2})$$

Recall that in the general SVM setting, one tries to minimize the hinge loss:

$$\ell_{\text{hinge}}(\mathbf{x}, y; \mathbf{w}) = [1 - y\mathbf{w}^T \mathbf{x}]_+ \quad (\text{A.3})$$

Equation A.2 can be interpreted as optimizing the effective loss function

$$\ell_{\text{hinge}}^{\text{rob}}(\mathbf{x}, y; \mathbf{w}) = \max_{\|\Delta \mathbf{x}\| \leq \delta} [1 - y\mathbf{w}^T (\mathbf{x} + \Delta \mathbf{x})]_+ \quad (\text{A.4})$$

We say that this loss function is robust, in the sense that it represents the worst-case loss subject to the potential action of the adversary.

2. Computing the optimal displacement

In order to derive a closed form for the loss function $\ell_{\text{hinge}}^{\text{rob}}$, we should explore the nature of the adversarial choice in our model. Intuitively, the adversary will try to relocate the point to the wrong side of the separating hyperplane. For this end, it is pointless to move the point along any axes not orthogonal to the separating hyperplane. This idea is visualized in Figure A.1. We will now prove this simple theorem:

Theorem 2.1: The optimum of the maximization in Equation A.4 is achieved at $\mathbf{x}_{\text{opt}} = \mathbf{x} - \delta \frac{\mathbf{w}}{\|\mathbf{w}\|}$

Proof: First we observe that the function $f(z) = [1 - z]_+$ is a monotone non-increasing function of its argument z . Thus, maximizing $f(z)$ is equivalent to minimizing z . By the Cauchy-Schwartz inequality, we have that $|y\mathbf{w}^T \Delta \mathbf{x}| \leq \|\mathbf{w}\| \cdot \|\Delta \mathbf{x}\|$, with equality iff $\Delta \mathbf{x}$ is proportional to \mathbf{w} . Therefore, the minimal value possible is attained at $\Delta \mathbf{x}_{\text{opt}} = -\delta \frac{\mathbf{w}}{\|\mathbf{w}\|}$. We conclude that $\mathbf{x}_{\text{opt}} = \mathbf{x} - \delta \frac{\mathbf{w}}{\|\mathbf{w}\|}$ as claimed. ■

Plugging the result of the theorem above into Equation A.4 we end up with

$$\ell_{\text{hinge}}^{\text{rob}}(\mathbf{x}, y; \mathbf{w}) = [1 - y\mathbf{w}^T \mathbf{x} + \delta \|\mathbf{w}\|]_+ \quad (\text{A.5})$$

3. ASVC: Adversarial Support Vector Classification

The fact that Equation A.4 has a simple closed-form solution allows us to employ the algorithmic scheme of alternating optimization for Equation A.1. The structure of the algorithm is quite simple:

1. Alternately:
 - (a) Optimize for \mathbf{w}
 - (b) Optimize for $\Delta \mathbf{x}^1, \Delta \mathbf{x}^2, \dots, \Delta \mathbf{x}^M$

Until convergence.

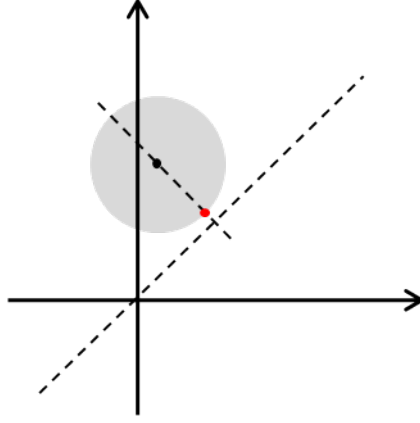


Figure A.1: The adversarial displacement employed by ASVC

Notice that 1a is nothing more than an SVM taking the displaced points as input. Furthermore, 1b has a closed-form solution as we have proved in Theorem 2.1. Thus, to solve for the optimal classifier, any off-the-shelf SVM solver can be used. We end up with Algorithm 5.

Algorithm 5: ASVC($\mathcal{S}, \delta, \lambda, T, k$)

Data: Training set \mathcal{S} , radius δ , tradeoff λ

Result: The weight vector w

$w \leftarrow 0$;

repeat

$\Delta x^m \leftarrow -\delta \frac{w}{\|w\|}$;
 $\tilde{\mathcal{S}} \leftarrow \{x^m + \Delta x^m\}_{x^m \in \mathcal{S}}$;
 $w \leftarrow \text{solveSVM}(\tilde{\mathcal{S}}, \lambda)$

until convergence ;

return w ;

4. The Multiclass Case

Pretty similar ideas can be adopted in order to generalize ASVC for the multiclass case.

The multi-hinge loss is defined as

$$\ell_{mult}(x^m, y^m; w_1, w_2, \dots, w_C) = \max_{y=1,2,\dots,C} [\delta_{y,y^m} - (w_{y^m} - w_y)^T x^m] \quad (\text{A.6})$$

Using the notions of the previous section, we define

$$\ell_{mult}^{\text{single}}(x^m, y^m; w_1, w_2, \dots, w_C) = \max_{\|\Delta x\| \leq \delta} \max_{y=1,2,\dots,C} [\delta_{y,y^m} - (w_{y^m} - w_y)^T (x^m + \Delta x)] \quad (\text{A.7})$$

Note the order of maximization can be changes, i.e.

$$\ell_{mult}^{\text{single}}(\mathbf{x}^m, y^m; \mathbf{w}_1, \mathbf{w}_2, \dots, \mathbf{w}_C) = \max_{y=1,2,\dots,C} \max_{\|\Delta \mathbf{x}\| \leq \delta} [\delta_{y,y^m} - (\mathbf{w}_{y^m} - \mathbf{w}_y)^T (\mathbf{x}^m + \Delta \mathbf{x})] \quad (\text{A.8})$$

Applying a slight variation of Theorem 2.1, we conclude with

$$\begin{aligned} \ell_{mult}^{\text{single}}(\mathbf{x}^m, y^m ; \mathbf{w}_1, \mathbf{w}_2, \dots, \mathbf{w}_C) \\ &= \max_{y=1,2,\dots,C} \left[\delta_{y,y^m} - (\mathbf{w}_{y^m} - \mathbf{w}_y)^T \left(\mathbf{x}^m - \delta \frac{\mathbf{w}_{y^m} - \mathbf{w}_y}{\|\mathbf{w}_{y^m} - \mathbf{w}_y\|} \right) \right] \\ &= \max_{y=1,2,\dots,C} [\delta_{y,y^m} - (\mathbf{w}_{y^m} - \mathbf{w}_y)^T \mathbf{x}^m + \delta \|\mathbf{w}_{y^m} - \mathbf{w}_y\|] \end{aligned}$$

5. Related work

Our ASVC algorithm is a mirror reflection of TSVC presented in (Bi & Zhang, NIPS04). TSVC performs alternating optimization, each time replacing the set of training samples with $\{\mathbf{x}^i + y^i \delta^i \frac{\mathbf{w}}{\|\mathbf{w}\|}\}$, which are more distant from the separator (thus, easier to classify). The idea there is to address the case in which noisy data distracts the classifier, by using the shifted training sets.

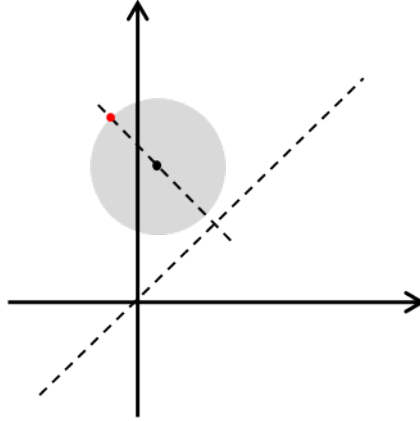


Figure A.2: The displacement employed by TSVC

Appendix B

Diagonal Covariance

In this appendix we discuss the case in which the adversary is constrained to choose a diagonal covariance matrix. This setting corresponds to the case when the noise is alligned to the primary axes. In this case we are able to give a closed form analytical result, subject to a bounded trace constraint on the covariance matrix.

The adversarial choice problem can be written

$$\max_{\Sigma=\text{diag}(a_1, a_2, \dots, a_d) \text{ } tr(\Sigma) \leq \beta} L(\mathbf{x}^m, y^m; \mathbf{w}, \mathbf{w}^T \Sigma \mathbf{w}) \quad (\text{B.1})$$

Let us expand

$$\begin{aligned} \mathbf{w}^T \Sigma \mathbf{w} &= \mathbf{w}^T \text{diag}(a_1, a_2, \dots, a_d) \mathbf{w} \\ &= \sum_{i=1}^d a_i w_i^2 = \mathbf{a}^T \mathbf{w}^{\cdot 2} \end{aligned}$$

where $\mathbf{w}^{\cdot 2}$ represents the coordinate-wise product of \mathbf{w} with itself. Let i^* be the index of the maximal entry in $\mathbf{w}^{\cdot 2}$. It hold that

$$\mathbf{w}^T \Sigma \mathbf{w} \leq \sum_i \beta_i w_{i^*}^2 \quad (\text{B.2})$$

Using the same argumentation as in Chapter 2, we conclude that the adversary will choose the covariance matrix

$$\Sigma^* = \beta \mathbf{e}_{i^* i^*} \quad (\text{B.3})$$

where \mathbf{e}_{ij} is the matrix having zeros in all of its entries beside (i, j) , where it takes the value 1. The geometric meaning of this result is that the adversary will choose to spread the noise in a single direction, along the primary axis that creates the biggest angle with the separating hyperplane.

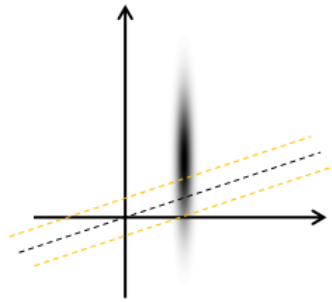


Figure B.1: Under the diagonal covariance restriction, the adversary will choose to spread the noise in a unique direction. This direction is the one that creates the biggest angle with the separating hyperplane.

Appendix C

Using the Multi-Hinge Loss

The most common generalization of the hinge loss for the multiclass case is the following loss function

$$\ell_{mult}(\mathbf{x}^m, y^m; \mathbf{w}_1, \dots, \mathbf{w}_C) = \max_y [\mathbf{w}_y^T \mathbf{x}^m - \mathbf{w}_{y^m}^T \mathbf{x}^m + \delta_{y,y^m}] \quad (\text{C.1})$$

(see Crammer and Singer [2002]). In this appendix we point out some of the issues that made us choose to work with the sum-of-hinges loss function and not with the one above.

If we plug the multi-hinge loss into our framework, we get the following learning problem:

$$\min_{\mathbf{w}} \sum_m \max_{\Sigma \in S} \int p(\hat{\mathbf{x}} | \mathbf{x}^m; \Sigma) \max_y [\mathbf{w}_y^T \hat{\mathbf{x}} - \mathbf{w}_{y^m}^T \hat{\mathbf{x}} + \delta_{y,y^m}] d\hat{\mathbf{x}} \quad (\text{C.2})$$

Define $\Delta \mathbf{w}_{y,y^m} = \mathbf{w}_y - \mathbf{w}_{y^m}$ and write:

$$\min_{\mathbf{w}} \sum_m \max_{\Sigma \in S} \int p(\hat{\mathbf{x}} | \mathbf{x}^m; \Sigma) \max_y [\Delta \mathbf{w}_{y,y^m}^T \hat{\mathbf{x}} + \delta_{y,y^m}] d\hat{\mathbf{x}} \quad (\text{C.3})$$

And for Gaussian noise this is:

$$\min_{\mathbf{w}} \sum_m \max_{\Sigma \in S} c|\Sigma|^{-0.5} \int e^{-\frac{1}{2} \mathbf{n}^T \Sigma^{-1} \mathbf{n}} \max_y [\Delta \mathbf{w}_{y,y^m}^T \mathbf{x}^m + \Delta \mathbf{w}_{y,y^m}^T \mathbf{n} + \delta_{y,y^m}] d\mathbf{n} \quad (\text{C.4})$$

The ability to understand the solution of the adversarial choice problem in this case, is connected to the ability to understand the expectation of the maximum of a set of normal random variables. This problem probably does not have an analytical solution (see Ross [2003]).

UNIDIRECTIONAL NOISE

In another approach we have studied, we assumed an adversary that spreads the noise in a single direction. The motivation for this kind of adversary is the solution to the adversarial choice problem in the binary case.

We formulate the problem by letting the adversary to choose a unit length vector. Thus, in the case of unidirectional noise, the task that the adversary faces is:

$$\max_{\mathbf{v}: \|\mathbf{v}\| \leq 1} \int_{\mathbb{R}} \mathcal{N}_z(0, \sigma^2) \max_y [\Delta \mathbf{w}_{y, y^m}^T \mathbf{x}^m + \Delta \mathbf{w}_{y, y^m}^T \mathbf{n} + \delta_{y, y^m}] dz \quad (\text{C.5})$$

The integrand (excluding the pdf) is a piecewise linear function. The knees of this function as well as the slopes of the linear sections are strongly dependent on \mathbf{v} . Nonetheless, it is impossible to find a closed form solution for the position of the knees. Therefore, we find this direction inapplicable in our case, as well.